DCT and DST Filtering with Sparse Graph Operators

Keng-Shih Lu, *Member, IEEE*, Antonio Ortega, *Fellow, IEEE*, Debargha Mukherjee, *Fellow, IEEE*, and Yue Chen

***Abstract*—Graph filtering is a fundamental tool in graph signal processing. Polynomial graph filters (PGFs), defined as polynomials of a fundamental graph operator, can be implemented in the vertex domain, and usually have a lower complexity than frequency domain filter implementations. In this paper, we focus on the design of filters for graphs with graph Fourier transform (GFT) corresponding to a discrete trigonometric transform (DTT), i.e., one of 8 types of discrete cosine transforms (DCT) and 8 discrete sine transforms (DST). In this case, we show that multiple sparse graph operators can be identified, which allows us to propose a generalization of PGF design: *multivariate polynomial graph filter (MPGF)*. First, for the widely used DCT-II (type-2 DCT), we characterize a set of sparse graph operators that share the DCT-II matrix as their common eigenvector matrix. This set contains the well-known connected line graph. These sparse operators can be viewed as graph filters operating in the DCT domain, which allows us to approximate any DCT graph filter by a MPGF, leading to a design with more degrees of freedom than the conventional PGF approach. Then, we extend those results to all of the 16 DTTs as well as their 2D versions, and show how their associated sets of multiple graph operators can be determined. We demonstrate experimentally that ideal low-pass and exponential DCT/DST filters can be approximated with higher accuracy with similar runtime complexity. Finally, we apply our method to transform-type selection in a video codec, AV1, where we demonstrate significant encoding time savings, with a negligible compression loss.**

***Index Terms*—graph filtering, discrete cosine transform, asymmetric discrete sine transform, graph Fourier transform**

I. Introduction

Graph signal processing (GSP) [1]–[3] extends classical signal processing concepts to data living on irregular domains. In GSP, the data domain is represented by a graph, and the measured data is called graph signal, where each signal sample corresponds to a graph vertex, and relations between samples are captured by the graph edges. Filtering, where frequency components of a signal are attenuated or amplified, is a fundamental operation in signal processing. Similar to conventional filters in digital signal processing, which manipulate signals in Fourier domain, a graph filter can be characterized by a frequency response that indicates how much the filter amplifies each graph frequency component. This notion of frequency selection leads to various applications, including graph signal denoising [4]–[6], classification [7] and clustering [8], and graph convolutional neural networks [9], [10].

For an undirected graph, a frequency domain graph filter operation $\mathbf{y} = \mathbf{H}\mathbf{x}$ with input signal $\mathbf{x}$ and filter matrix

$$\mathbf{H} = \mathbf{\Phi} \cdot h(\mathbf{\Lambda}) \cdot \mathbf{\Phi}^\top, \quad h(\mathbf{\Lambda}) := \text{diag}(h(\lambda_1), \cdots, h(\lambda_N)) \tag{1}$$

involves a forward graph Fourier transform (GFT) $\mathbf{\Phi}^\top$, a frequency selective scaling operation $h(\mathbf{\Lambda})$, and an inverse GFT $\mathbf{\Phi}$. However, as fast GFT algorithms are only known for graphs with certain structural properties [11], the GFT can introduce a high computational overhead when the graph is arbitrary. To address this issue, graph filters can be implemented with polynomial operations in vertex domain:

$$\mathbf{H} = \sum_{k=0}^{K} g_k \mathbf{Z}^k, \quad \text{with } \mathbf{Z}^0 = \mathbf{I}, \tag{2}$$

where the g_k's are coefficients and $\mathbf{Z}$ is called the *graph shift operator*, *fundamental graph operator*, or *graph operator* for short. With this expression, graph filtering can be applied in the vertex (sample) domain via $\mathbf{y} = \mathbf{H}\mathbf{x}$, which does not require GFT computations. A graph filter in the form of (2) is usually called an *FIR graph filter* [12], [13] as it can be viewed as an analogy to conventional FIR filters with order K, which are polynomials of the delay z. In this paper, we call the filters defined in (2) *polynomial graph filters (PGFs)*.

Various methods for designing vertex domain graph filters given a desired frequency response have been studied in the literature. Least squares design of polynomial filters given a target frequency response was introduced in [14]–[16]. The recurrence relations of Chebyshev polynomials provide computational benefits in distributed filter implementations as shown in [17], [18]. In [19] an extension of graph filter operations to a node-variant setting is proposed, along with polynomial approximation approaches using convex optimization. Autoregressive moving average (ARMA) graph filters, whose frequency responses are characterized by rational polynomials, have been investigated in [12], [20] in both static and time-varying settings. Design strategies of ARMA filters are further studied in [21], which provides comparisons to PGFs. Furthermore, in [13], state-of-the-art filtering methods have been extended to an edge-variant setting. All these methods are based on using a single graph operator $\mathbf{Z}$.

The possibility of using *multiple operators* was first observed in [22]. Multiple graph operators $\mathcal{Z} = \{\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)}, \ldots, \mathbf{Z}^{(m)}\}$ that are jointly diagonalizable (i.e., have a common eigenbasis) can be obtained for both cycle graphs [22] and line graphs [23]. Essentially, those operators are by themselves graph filter matrices with different frequency responses. Thus, unlike (2), which is a polynomial of a single operator, we can design graph filters of the form:

$$\mathbf{H}_{\mathcal{Z},K} = p_K(\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)}, \ldots, \mathbf{Z}^{(m)}), \tag{3}$$

where $p_K(\cdot)$ stands for a multivariate polynomial with degree K and arbitrary coefficients. Given the graph filter expression (3), iterative algorithms for filter implementation have been recently studied in [24]. Since $\mathbf{H}_{\{\mathbf{Z}\},K} = p_K(\mathbf{Z})$ reduces to (2), the form (3) is a generalization of the PGF expression. We refer to (3) as *multivariate polynomial graph filter (MPGF)*.

Filtering operations based on the well-known discrete cosine transform (DCT) and discrete sine transform (DST) [25], [26], as well as their extension to all discrete trigonometric transforms (DTTs), i.e., 8 types of DCTs and 8 types of DSTs [27], are referred to as DTT filters. All DTTs are GFTs associated with uniform line graphs [26], [27] and are based on the following operations: 1) computing the DTT of the input signal, 2) scaling each of the computed DTT coefficients, and 3) performing the inverse DTT. Applications of DTT filters include image resizing [28], biomedical signal processing [29], medical imaging [30], and video coding [31].

While DTTs have long been studied and are among the most widely used transforms, sample domain DTT filtering operation have not been studied yet, in contrast with DFT filters for which have

K.-S. Lu, D. Mukherjee and Y. Chen are with Google, Mountain View, CA 94043, USA (email: kslu@google.com; debargha@google.com; yuec@google.com). Most of this work has been done while K.-S. Lu was a PhD student at USC.

A. Ortega is with the Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA 90089, USA (email: ortega@sipi.usc.edu).

This work was supported in part by a Taiwan Fellowship, by the US National Science Foundation (CCF-1527874 and CCF-1410009) and by a research grant from Google.

FIR implementations are known. This motivates our study of graph-based approaches to design and implement DTT filters. The main advantage of graph based approaches is that they do not require the DTT and inverse DTT steps, and instead can be applied directly in the signal domain, using suitable graph operators. In this work, we will focus on the design of efficient sample domain (graph vertex domain) DTT filters. Specifically, for GFTs corresponding to any of the 16 DTTs, we derive a family $\mathcal{Z}$ of sparse graph operators with closed form expressions, which can be used in addition to the graph operator obtained from the well-known line graph model [27]. In this way, efficient DTT filters can be obtained using PGF and MPGF design approaches, yielding a lower complexity than a DTT filter implementation in the transform domain. Our main contributions are summarized as follows:

1) We introduce multiple sparse graph operators specific to DTTs and allowing fast MPGF implementations. These sparse operators are DTT filters, which are special cases of graphs filters, but have not been considered in the general graph filtering literature [12], [13], [18]–[21]. While [26] and [27] establish the connection between DTTs and line graphs, our proposed sparse graph operators for DTTs, which are no longer restricted to be line graphs, had not been studied in the literature.
2) We introduce novel DTT filter design methods for graph vertex domain implementation. While in related work [27], [32]–[34], DTT filtering is typically performed in the transform domain using convolution-multiplication properties, we introduce sample domain DTT filter implementations based on PGF and MPGF designs, and show that our designs with low degree polynomials lead to faster implementations as compared to those designs that require forward and inverse DTTs, especially in cases where DTT size is large.
3) In addition to the well-known least squares graph filter design, we propose a novel minimax design approach for both PGFs and MPGFs, which optimally minimizes the approximation error in terms of maximum absolute error in the graph frequencies. Compared to the recent work in [35], which considers a minimax criterion on the discrete frequency domain $[0, \pi]$ mapped from the normalized graph frequency domain $[0, 2]$, we simply define the minimax error on the graph frequency domain, and provide a design method with a lower complexity than that in [35].
4) We show that MPGFs could lead to more efficient DTT filter implementations, as compared to conventional PGF designs. A typical challenge for MPGF is the identification of multiple operators for an arbitrary graph. In this work, by introducing closed form expressions of the DTT operators, MPGFs for DTTs can be easily designed. In addition, MPGFs allow more flexible DTT filter designs than PGFs, which could benefit existing DTT-related applications. For example, we demonstrate that MPGFs are suitable for DTT filters with frequency responses that are non-smooth (e.g., ideal low-pass filters) or non-monotonic (e.g., bandpass filters). We also demonstrate experimentally the benefits of sparse DTT operators in image and video compression applications. In addition to filter operation, our approach can also be used to evaluate the transform domain weighted energy given by the Laplacian quadratic form, which has been used for rate-distortion optimization in the context of image and video coding [36], [37]. Following our recent work [23], we implement the proposed method in AV1, a real-world codec, where our method provides a speedup in the transform type search procedure.

We highlight that, while [24] studies MPGFs with a focus on distributed filter implementations, it does not investigate design approaches of MPGFs or how sparse operators for generic graphs can be obtained other than cycle and Cartesian product graphs. Our work complements the study in [24] by considering 1) the case where GFT is a DTT, which corresponds to various line graphs, and 2) techniques to design MPGFs. In addition, the work presented in this paper is a more general framework than our prior work in [23], since the Laplacian quadratic form operation used in [23] can be viewed as a special case of graph filtering operation. Furthermore, while our work in [23] was restricted to DCT/ADST, in this paper we have extended these ideas to all DTTs. The graph filters we study are also fundamentally different from those considered in [38], where different graph shifts characterize feature-wise similarity of different data samples, and they are not required to share a common eigenbasis.

The rest of this paper is organized as follows. We review graph filtering concepts and some relevant properties of DTTs in Sec. II. In Sec. III, we consider sparse operators for DTTs that can be obtained by extending well-known properties of DTTs. In Sec. IV we discuss operators for the DFT and the Hadamard transform, and provide some remarks on sparse operators for general graphs. Sec. V introduces PGF and MPGF design approaches using least squares and minimax criteria. Sec. VI shows distributed implementation for PGF and MPGF based on sparse operators as well as an efficient filter design for Laplacian quadratic form approximation. Experimental results are shown in Sec. VII to demonstrate the effectiveness of our methods in graph filter design as well as applications in video coding. Conclusions are given in Sec. VIII.

II. Preliminaries

We start by reviewing relevant concepts in graph signal processing and DTTs. In what follows, entries in a matrix that are not displayed are meant to be zero. Thus, the order-reversal permutation matrix is:

$$\mathbf{J} = \begin{pmatrix} & & & 1 \\ & & 1 & \\ & \iddots & & \\ 1 & & & \end{pmatrix},$$

which satisfies $\mathbf{J}^\top = \mathbf{J}$ and $\mathbf{J}\mathbf{J} = \mathbf{I}$. The transpose, Hermitian, and pseudo-inverse of matrix $\mathbf{A}$ are denoted as $\mathbf{A}^\top$, $\mathbf{A}^H$, and $\mathbf{A}^\dagger$, respectively.

A. Graph Fourier Transforms

Let $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathbf{W})$ be an undirected graph with N nodes and let $\mathbf{x}$ be a length-N graph signal associated to $\mathcal{G}$. Each node of $\mathcal{G}$ corresponds to an entry of $\mathbf{x}$, and each edge $e_{ij} \in \mathcal{E}$ describes the inter-sample relation between nodes i and j. The (i, j) entry of the weight matrix, $w_{i,j}$, is the weight of the edge e_{ij}, and $\theta_i := w_{i,i}$ is the weight of the self-loop on node i. Defining $\mathbf{\Theta} = \text{diag}(\theta_1, \ldots, \theta_N)$ and $\mathbf{D} = \text{diag}(d_1, \ldots, d_N)$ as diagonal matrices of self-loop weights and node degrees, $d_i = \sum_{j=1}^{N} w_{i,j}$, respectively, the unnormalized and normalized graph Laplacian matrices are

$$\mathbf{L} = \mathbf{D} - \mathbf{W} + \mathbf{\Theta}, \quad \boldsymbol{\mathcal{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}. \tag{4}$$

In what follows, unless stated otherwise, we refer to the unnormalized version, $\mathbf{L}$, as the graph Laplacian. All graphs we consider are assumed to be undirected.

The graph Fourier transform (GFT) is obtained from the eigen-decomposition of the graph Laplacian, $\mathbf{L} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\top$, with eigenvalues $\lambda_1 \leq \cdots \leq \lambda_N$ in ascending order. The vector of GFT coefficients for graph signal $\mathbf{x}$ is $\hat{\mathbf{x}} = \mathbf{\Phi}^\top\mathbf{x}$. We note that the variation of signal $\mathbf{x}$ on

the graph can be measured by the Laplacian quadratic form:

$$\mathbf{x}^\top \mathbf{L}\mathbf{x} = \sum_{(i,j)\in\mathcal{E}} w_{i,j}(x_i - x_j)^2 + \sum_{k=1}^{N} \theta_k x_k^2. \quad (5)$$

The columns of $\mathbf{\Phi}$, $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_N$ form an orthogonal basis and each of them can be viewed as a graph signal with variation equal to the associated eigenvalues $\lambda_1, \ldots, \lambda_N$, which are called *graph frequencies*.

B. Graph Filters

We consider a 1-hop graph operator $\mathbf{Z}$, which could be the adjacency matrix or one of the Laplacian matrices. For a given signal $\mathbf{x}$, $\mathbf{y} = \mathbf{Z}\mathbf{x}$ defines an operation where the output at each node is a function of values at its 1-hop neighbors (e.g., when $\mathbf{Z} = \mathbf{A}$, $y(i) = \sum_{j\in\mathcal{N}(i)} x(j)$, where $\mathcal{N}(i)$ is the set of nodes that are neighbors of i). Furthermore, it can be shown that $\mathbf{y} = \mathbf{Z}^K\mathbf{x}$ is a K-hop operation, and thus for a degree-K polynomial of $\mathbf{Z}$, as in (2), the output at node i depends on its K-hop neighbors. The operation in (2) is thus called a *graph filter*, an FIR graph filter, or a polynomial graph filter (PGF). In what follows, we refer to $\mathbf{Z}$ as *graph operator* for short[1]. For the rest of this paper, we choose $\mathbf{Z} = \mathbf{L}$ or define $\mathbf{Z}$ as a matrix with the same eigenbasis as $\mathbf{L}$, e.g., $\mathbf{Z}$ could be a polynomial of $\mathbf{L}$ such as $\mathbf{Z} = 2\mathbf{I} - \mathbf{L}$.

The matrix $\mathbf{\Phi}$ of eigenvectors of $\mathbf{Z} = \mathbf{L}$ is also the eigenvector matrix of any polynomial $\mathbf{H}$ in the form of (2). The eigenvalue $h(\lambda_j)$ of $\mathbf{H}$ associated to $\boldsymbol{\phi}_j$ is called the *frequency response* of λ_j. Note that with $\mathbf{y} = \mathbf{H}\mathbf{x}$, in the GFT domain we have $\hat{\mathbf{y}} = h(\mathbf{\Lambda})\hat{\mathbf{x}}$, meaning that the filter operator scales the signal component with λ_j frequency by $h(\lambda_j)$ in the GFT domain. We also note that (1) generalizes the notion of digital filter: when $\mathbf{\Phi}$ is the discrete Fourier transform (DFT) matrix, $\mathbf{H}$ reduces to the classical Fourier filter [15]. Given a desired graph frequency response $\mathbf{h} = (h_1, \ldots, h_N)^\top$, its associated polynomial coefficients in (2) can be obtained by solving a least squares minimization problem [15]:

$$\mathbf{g} = \underset{\mathbf{g}}{\operatorname{argmin}} \quad \|\mathbf{h} - \mathbf{\Psi}\mathbf{g}\|^2, \quad \text{where } \mathbf{\Psi} = \begin{pmatrix} 1 & \lambda_1 & \ldots & \lambda_1^K \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \lambda_N & \ldots & \lambda_N^K \end{pmatrix}, \quad (6)$$

with λ_j being the j-th eigenvalue of $\mathbf{Z}$. The PGF operation $\mathbf{y} = \mathbf{H}\mathbf{x}$ can be implemented efficiently by computing: 1) $\mathbf{t}^{(0)} = g_K\mathbf{x}$, 2) $\mathbf{t}^{(i)} = \mathbf{Z}\mathbf{t}^{(i-1)} + g_{K-i}\mathbf{x}$, and 3) $\mathbf{y} = \mathbf{t}^{(K)}$. This algorithm does not require GFT computation, and its complexity depends on the degree K and how sparse $\mathbf{Z}$ is (with lower complexity for sparser $\mathbf{Z}$).

C. Discrete Cosine and Sine Transforms

The discrete cosine transform (DCT) and discrete sine transform (DST) are orthogonal transforms that operate on a finite vector, with basis functions derived from cosines and sines, respectively. Discrete trigonometric transforms (DTTs) comprise eight types of DCT and eight types of DST, which are defined depending on how samples are taken from continuous cosine and sine functions [27], [39]. We denote them by DCT-I to DCT-VIII, and DST-I to DST-VIII, and list their forms in Table I.

DCT and ADST are widely used in image and video coding. In this paper, we refer the terms "DCT" and "ADST" to DCT-II and DST-IV[2], respectively, unless stated otherwise. For $j = 1, \ldots, N$ and $k = 1, \ldots, N$, we denote the k-th element of the j-th length-N DCT and ADST functions as

$$\text{DCT-II:} \quad u_j(k) = \sqrt{\frac{2}{N}} c_j \cos\frac{(j-1)(k-\frac{1}{2})\pi}{N}, \quad (7)$$

$$\text{DST-IV:} \quad v_j(k) = \sqrt{\frac{2}{N}} \sin\frac{(j-\frac{1}{2})(k-\frac{1}{2})\pi}{N}. \quad (8)$$

with normalization constant c_j being $1/\sqrt{2}$ for $j = 1$ and 1 otherwise. If those basis functions are written in vector form $\mathbf{u}_j, \mathbf{v}_j \in \mathbb{R}^N$, it was pointed out in [26] that the $\mathbf{u}_j$ are eigenvectors of the Laplacian matrix $\mathbf{L}_\mathrm{D}$, and in [27] that $\mathbf{v}_j$ are eigenvectors of $\mathbf{L}_\mathrm{A}$, with

$$\mathbf{L}_\mathrm{D} = \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix}, \quad \mathbf{L}_\mathrm{A} = \begin{pmatrix} 3 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix}. \quad (9)$$

This means that the DCT and ADST are GFTs corresponding to Laplacian matrices $\mathbf{L}_\mathrm{D}$ and $\mathbf{L}_\mathrm{A}$, respectively. Their associated graphs $\mathcal{G}_\mathrm{D}$ and $\mathcal{G}_\mathrm{A}$ with $N = 8$ are shown in the second subfigure in Fig. 1(a) and the fourth subfigure in Fig. 1(b), respectively. The eigenvalues of $\mathbf{L}_\mathrm{D}$ corresponding to $\mathbf{u}_j$ are $\omega_j = 2 - 2\cos((j-1)\pi/N)$, and those of $\mathbf{L}_\mathrm{A}$ corresponding to $\mathbf{v}_j$ are $\delta_j = 2 - 2\cos((j-1/2)\pi/N)$.

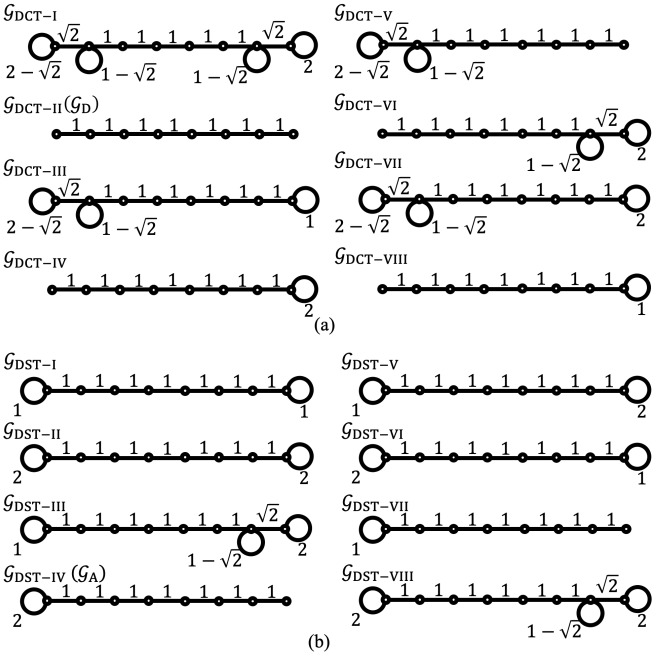


Fig. 1. Graphs associated to 8-point (a) DCTs and (b) DSTs.

The graphs corresponding to all 16 DTTs with length 8 are shown in Fig. 1. We note that all those graphs are uniform line graphs with different edge and self-loop patterns at the left and right ends. Thus, line graphs associated with arbitrary length DTT can be easily extended from those in Fig. 1. For instance, length 16 DST-I would correspond to a line graph with 16 nodes and two self-loops at its two end nodes, with all weights being 1. We also highlight that, in the scope of this paper, while graph Laplacians are required to be positive semi-definite, we allow negative edge or self-loop weights, such as $1 - \sqrt{2}$ in some of those graphs, since the main role of the graph operators is to provide efficient vertex domain filter implementation.

III. Sparse DCT and DST Operators

Classical PGFs can be extended to MPGFs [24] if multiple graph operators are available [22]. Let $\mathbf{L} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\top$ be a Laplacian with GFT

[1] In the literature, $\mathbf{Z}$ is often called *graph shift operator* [1], [12], [19], [22]. Here, we simply call it graph operator, since its properties are different from shift in conventional signal processing, which is always reversible, while the graph operator $\mathbf{Z}$, in most cases, is not.

[2] DST-VII was shown to optimally decorrelate intra residual pixels under a Gaussian Markov model [40], [41], but its variant DST-IV is amenable to fast implementations while experimentally achieving a similar coding efficiency [42]. In this paper, we refer to DST-IV as ADST, as in the AV1 codec [43].

$\mathbf{\Phi}$ and assume we have a series of graph operators $\mathcal{Z} = \{\mathbf{Z}^{(k)}\}_{k=1}^{M}$ that share the same eigenvectors as $\mathbf{L}$, but with different eigenvalues:

$$\mathbf{Z}^{(k)} = \mathbf{\Phi}\mathbf{\Lambda}^{(k)}\mathbf{\Phi}^{\top}, \quad \mathbf{\Lambda}^{(k)} = \mathrm{diag}(\boldsymbol{\lambda}^{(k)}) = \mathrm{diag}(\lambda_1^{(k)}, \ldots, \lambda_N^{(k)}),$$

where $\boldsymbol{\lambda}^{(k)} = (\lambda_1^{(k)}, \ldots, \lambda_N^{(k)})^{\top}$ denotes the vector of eigenvalues of $\mathbf{Z}^{(k)}$. When the polynomial degree is $K = 1$ in (3), we have:

$$\mathbf{H}_{\mathcal{Z},1} = g_0\mathbf{I} + \sum_{m=1}^{M} g_m\mathbf{Z}^{(m)}, \tag{10}$$

where g_k are coefficients. When $K = 2$, we have

$$\begin{aligned}\mathbf{H}_{\mathcal{Z},2} = g_0\mathbf{I} &+ \sum_{m=1}^{M} g_m\mathbf{Z}^{(m)} \\ &+ g_{M+1}\mathbf{Z}^{(1)}\mathbf{Z}^{(1)} + g_{M+2}\mathbf{Z}^{(1)}\mathbf{Z}^{(2)} + \cdots + g_{2M}\mathbf{Z}^{(1)}\mathbf{Z}^{(M)} \\ &+ g_{2M+1}\mathbf{Z}^{(2)}\mathbf{Z}^{(2)} + \cdots + g_{3M-1}\mathbf{Z}^{(2)}\mathbf{Z}^{(M)} \\ &+ \ldots \\ &+ g_{(M^2+3M)/2}\mathbf{Z}^{(M)}\mathbf{Z}^{(M)},\end{aligned} \tag{11}$$

where the terms $\mathbf{Z}^{(j)}\mathbf{Z}^{(i)}$ with $j > i$ are not required in (11) because all operators commute, i.e., $\mathbf{Z}^{(i)}\mathbf{Z}^{(j)} = \mathbf{Z}^{(j)}\mathbf{Z}^{(i)}$. Expressions with a higher degree can be obtained with polynomial kernel expansion [44]. We also note that, since $\mathbf{H}_{\{\mathbf{Z}\},K}$ reduces to the form of $\mathbf{H}$ in (2), $\mathbf{H}_{\mathcal{Z},K}$ is a generalization of PGF and thus provides more degrees of freedom for the filter design procedure.

As pointed out in the introduction, DTT filters are essentially graph filters. This means that they can be implemented with PGFs as in (2), without applying any forward or inverse DTT. Next, we will go one step further by introducing multiple sparse operators for each DTT, which allows the implementation of DTT filters using MPGFs.

The use of polynomial (2) to perform filtering in the vertex domain, rather in the frequency domain, is advantageous only if the operator is sparse. In this section, our main goal is to find multiple *sparse* DTT operators, which have only one or two nonzero elements per row and thus are the most sparse ones among all possible DTT operators. While other useful operators with more nonzero elements (e.g., with 3 or 4 nonzero elements per row) may exist, they can possibly be expressed as linear combinations of the sparsest ones. Thus, the proposed MPGF design in later sections will be based on the sparsest operators.

In what follows, the result of [26] will be first generalized in Sec. III-A to derive multiple sparse operators from a single operator for DCT-II. A toy example for those operators is provided in Sec. III-B. Then, in Sec. III-C, we further show that, in addition to DCT-II, operators can be derived for all 16 DTTs based on the approach in Sec. III-A. Finally, sparse operators associated to 2D DTTs are presented in Sec. III-D.

A. Sparse DCT-II Operators

Let $\mathbf{u}_j$ denote the DCT basis vector with entries from (7) and let $\mathbf{L}_{\mathrm{D}}$ be the Laplacian of a uniform line graph, (9). The following proposition from [26] and its proof, developed for the line graph case, will be useful to find additional sparse operators:

Proposition 1 ([26])**.** *$\mathbf{u}_j$ is an eigenvector of $\mathbf{L}_D$ with eigenvalue $\omega_j = 2 - 2\cos((j-1)\pi/N)$ for each $j = 1, \ldots N$*

Proof: It suffices to show an equivalent equation: $\mathbf{Z}_{\text{DCT-II}} \cdot \mathbf{u}_j = (2 - \omega_j)\mathbf{u}_j$, where

$$\mathbf{Z}_{\text{DCT-II}} = 2\mathbf{I} - \mathbf{L}_{\mathrm{D}} = \begin{pmatrix} 1 & 1 & & & \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ & & & 1 & 1 \end{pmatrix}. \tag{12}$$

TABLE I
DEFINITIONS OF DTTS AND THE EIGENVALUES OF THEIR SPARSE OPERATORS. THE INDICES j AND k RANGE FROM 1 TO N. SCALING FACTORS ARE GIVEN BY $c_j = 1/\sqrt{2}$ FOR $j = 1$ AND $c_j = 1$ OTHERWISE; $d_j = 1/\sqrt{2}$ FOR $j = N$ AND $d_j = 1$ OTHERWISE.

DTT	Transform functions $\phi_j(k)$	Eigenvalue of $\mathbf{Z}^{(\ell)}$ associated to $\boldsymbol{\phi}_j$
DCT-I	$\sqrt{\frac{2}{N-1}}c_j c_k d_j d_k \cos\frac{(j-1)(k-1)\pi}{N-1}$	$2\cos\left(\frac{\ell(j-1)\pi}{N-1}\right)$
DCT-II	$\sqrt{\frac{2}{N}}c_j \cos\frac{(j-1)(k-1/2)\pi}{N}$	$2\cos\left(\frac{\ell(j-1)\pi}{N}\right)$
DCT-III	$\sqrt{\frac{2}{N}}c_k \cos\frac{(j-1/2)(k-1)\pi}{N}$	$2\cos\left(\frac{\ell(j-1/2)\pi}{N}\right)$
DCT-IV	$\sqrt{\frac{2}{N}}\cos\frac{(j-1/2)(k-1/2)\pi}{N}$	$2\cos\left(\frac{\ell(j-1/2)\pi}{N}\right)$
DCT-V	$\frac{2}{\sqrt{2N-1}}c_j c_k \cos\frac{(j-1)(k-1)\pi}{N-1/2}$	$2\cos\left(\frac{\ell(j-1)\pi}{N-1/2}\right)$
DCT-VI	$\frac{2}{\sqrt{2N-1}}c_j d_k \cos\frac{(j-1)(k-1/2)\pi}{N-1/2}$	$2\cos\left(\frac{\ell(j-1)\pi}{N-1/2}\right)$
DCT-VII	$\frac{2}{\sqrt{2N-1}}d_j c_k \cos\frac{(j-1/2)(k-1)\pi}{N-1/2}$	$2\cos\left(\frac{\ell(j-1/2)\pi}{N-1/2}\right)$
DCT-VIII	$\frac{2}{\sqrt{2N+1}}\cos\frac{(j-1/2)(k-1/2)\pi}{N+1/2}$	$2\cos\left(\frac{\ell(j-1/2)\pi}{N+1/2}\right)$
DST-I	$\sqrt{\frac{2}{N+1}}\sin\frac{jk\pi}{N+1}$	$2\cos\left(\frac{\ell j\pi}{N+1}\right)$
DST-II	$\sqrt{\frac{2}{N}}d_j \sin\frac{j(k-1/2)\pi}{N}$	$2\cos\left(\frac{\ell j\pi}{N}\right)$
DST-III	$\sqrt{\frac{2}{N}}d_k \sin\frac{(j-1/2)k\pi}{N}$	$2\cos\left(\frac{\ell(j-1/2)\pi}{N}\right)$
DST-IV	$\sqrt{\frac{2}{N}}\sin\frac{(j-1/2)(k-1/2)\pi}{N}$	$2\cos\left(\frac{\ell(j-1/2)\pi}{N}\right)$
DST-V	$\frac{2}{\sqrt{2N+1}}\sin\frac{jk\pi}{N+1/2}$	$2\cos\left(\frac{\ell j\pi}{N+1/2}\right)$
DST-VI	$\frac{2}{\sqrt{2N+1}}\sin\frac{j(k-1/2)\pi}{N+1/2}$	$2\cos\left(\frac{\ell j\pi}{N+1/2}\right)$
DST-VII	$\frac{2}{\sqrt{2N+1}}\sin\frac{(j-1/2)k\pi}{N+1/2}$	$2\cos\left(\frac{\ell(j-1/2)\pi}{N+1/2}\right)$
DST-VIII	$\frac{2}{\sqrt{2N-1}}d_j d_k \sin\frac{(j-1/2)(k-1/2)\pi}{N-1/2}$	$2\cos\left(\frac{\ell(j-1/2)\pi}{N-1/2}\right)$

For $1 \le p \le N$, the p-th element of $\mathbf{Z}_{\text{DCT-II}} \cdot \mathbf{u}_j$ is

$$(\mathbf{Z}_{\text{DCT-II}} \cdot \mathbf{u}_j)_p = \begin{cases} u_j(1) + u_j(2), & p = 1 \\ u_j(p-1) + u_j(p+1), & 2 \le p \le N-1 \\ u_j(N-1) + u_j(N), & p = N. \end{cases}$$

Following the expression in (7), we extend the definition of $u_j(k)$ to an arbitrary integer k. The even symmetry of the cosine function at 0 and π gives $u_j(0) = u_j(1)$ and $u_j(N) = u_j(N+1)$, and thus

$$\begin{aligned}(\mathbf{Z}_{\text{DCT-II}} \cdot \mathbf{u}_j)_1 &= u_j(0) + u_j(2), \\ (\mathbf{Z}_{\text{DCT-II}} \cdot \mathbf{u}_j)_N &= u_j(N-1) + u_j(N+1).\end{aligned} \tag{13}$$

This means that for all $p = 1, \ldots, N$,

$$(\mathbf{Z}_{\text{DCT-II}} \cdot \mathbf{u}_j)_p = u_j(p-1) + u_j(p+1) \tag{14a}$$

$$= \sqrt{\frac{2}{N}}c_j\left[\cos\frac{(j-1)(p-\frac{3}{2})\pi}{N} + \cos\frac{(j-1)(p+\frac{1}{2})\pi}{N}\right] \tag{14b}$$

$$= 2\sqrt{\frac{2}{N}}c_j \cos\frac{(j-1)(p-\frac{1}{2})\pi}{N}\cos\frac{(j-1)\pi}{N} \tag{14c}$$

$$= (2-\omega_j)u_j(p), \tag{14d}$$

which verifies $\mathbf{Z}_{\text{DCT-II}} \cdot \mathbf{u}_j = (2-\omega_j)\mathbf{u}_j$. Note that in (14b), we have applied the sum-to-product trigonometric identity:

$$\cos\alpha + \cos\beta = 2\cos\left(\frac{\alpha+\beta}{2}\right)\cos\left(\frac{\alpha-\beta}{2}\right). \quad \square \tag{15}$$

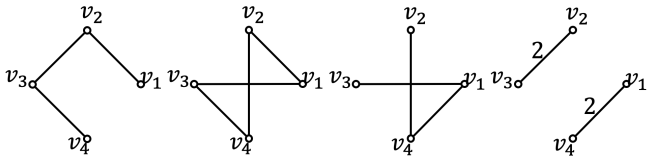
Now we can extend the above result as follows. When $u_j(q \pm 1)$ is replaced by $u_j(q \pm \ell)$ in (14a), this identity also applies, which

$$\begin{pmatrix}1&1&0&0\\1&0&1&0\\0&1&0&1\\0&0&1&1\end{pmatrix}\begin{pmatrix}0&1&1&0\\1&0&0&1\\1&0&0&1\\0&1&1&0\end{pmatrix}\begin{pmatrix}0&0&1&1\\0&1&0&1\\1&0&1&0\\1&1&0&0\end{pmatrix}\begin{pmatrix}0&0&0&2\\0&0&2&0\\0&2&0&0\\2&0&0&0\end{pmatrix}$$

(a) $\mathbf{Z}^{(1)}_{\text{DCT-II}}$, $\mathbf{Z}^{(2)}_{\text{DCT-II}}$, $\mathbf{Z}^{(3)}_{\text{DCT-II}}$, and $\mathbf{Z}^{(4)}_{\text{DCT-II}}$

$$\begin{pmatrix}1&-1&0&0\\-1&2&-1&0\\0&-1&2&-1\\0&0&-1&1\end{pmatrix}\begin{pmatrix}2&-1&-1&0\\-1&2&0&-1\\-1&0&2&-1\\0&-1&-1&2\end{pmatrix}\begin{pmatrix}2&0&-1&-1\\0&1&0&-1\\-1&0&1&0\\-1&-1&0&2\end{pmatrix}\begin{pmatrix}2&0&0&-2\\0&2&-2&0\\0&-2&2&0\\-2&0&0&2\end{pmatrix}$$

(b) $\mathbf{L}^{(1)}_{\text{DCT-II}}$, $\mathbf{L}^{(2)}_{\text{DCT-II}}$, $\mathbf{L}^{(3)}_{\text{DCT-II}}$, and $\mathbf{L}^{(4)}_{\text{DCT-II}}$



(c) $\mathcal{G}^{(1)}_{\text{DCT-II}}$, $\mathcal{G}^{(2)}_{\text{DCT-II}}$, $\mathcal{G}^{(3)}_{\text{DCT-II}}$, and $\mathcal{G}^{(4)}_{\text{DCT-II}}$

Fig. 2. (a) Sparse operators $\mathbf{Z}^{(j)}_{\text{DCT-II}}$, (b) their associated Laplacian matrices $\mathbf{L}^{(j)}_{\text{DCT-II}} = 2\mathbf{I} - \mathbf{Z}^{(j)}_{\text{DCT-II}}$, and (c) associated graphs $\mathcal{G}^{(j)}$ for the length-4 DCT-II.

generalizes (14a)-(14d) to

$$\begin{aligned}
&u_j(p-\ell) + u_j(p+\ell)\\
&= \sqrt{\frac{2}{N}}c_j\left[\cos\frac{(j-1)(p-\ell-\frac{1}{2})\pi}{N} + \cos\frac{(j-1)(p+\ell-\frac{1}{2})\pi}{N}\right]\\
&= 2\sqrt{\frac{2}{N}}c_j\cos\frac{(j-1)(p-\frac{1}{2})\pi}{N}\cos\frac{\ell(j-1)\pi}{N}\\
&= \left(2\cos\frac{\ell(j-1)\pi}{N}\right)u_j(p). \qquad (16)
\end{aligned}$$

As in (13), we can apply even symmetry of the cosine function at 0 and π, to replace indices $p-\ell$ or $p+\ell$ that are out of the range $[1, N]$ by those within the range:

$$\begin{aligned}
u_j(p-\ell) &= u_j(-p+\ell+1),\\
u_j(p+\ell) &= u_j(-p-\ell+2N+1).
\end{aligned}$$

Then, an $N \times N$ matrix $\mathbf{Z}^{(\ell)}_{\text{DCT-II}}$ can be defined such that the left hand side of (16) corresponds to $(\mathbf{Z}^{(\ell)}_{\text{DCT-II}} \cdot \mathbf{u}_j)_p$. This leads to the following proposition:

Proposition 2. *For $\ell = 1, \ldots, N-1$, we define $\mathbf{Z}^{(\ell)}_{DCT\text{-}II}$ as a $N \times N$ matrix, whose p-th row has only two non-zero elements specified as follows:*

$$\left(\mathbf{Z}^{(\ell)}_{DCT\text{-}II}\right)_{p,q_1} = 1, \quad q_1 = \begin{cases} p-\ell, & \text{if } p-\ell \geq 1\\ -p+\ell+1, & \text{otherwise}\end{cases}$$

$$\left(\mathbf{Z}^{(\ell)}_{DCT\text{-}II}\right)_{p,q_2} = 1, \quad q_2 = \begin{cases} p+\ell, & \text{if } p+\ell \leq N\\ -p-\ell+2N+1, & \text{otherwise}\end{cases}$$

This matrix $\mathbf{Z}^{(\ell)}_{DCT\text{-}II}$ has eigenvectors $\mathbf{u}_j$ with associated eigenvalues $2\cos(\ell(j-1)\pi/N)$ for $j = 1, \ldots, N$.

Note that $\mathbf{Z}^{(1)}_{\text{DCT-II}} = \mathbf{Z}_{\text{DCT-II}}$ as in (12). Taking $\ell = 2$ and $\ell = 3$ and following Proposition 2, we see that nonzero elements in $\mathbf{Z}^{(2)}_{\text{DCT-II}}$ and $\mathbf{Z}^{(3)}_{\text{DCT-II}}$ form rectangle-like patterns similar to that in $\mathbf{Z}_{\text{DCT-II}}$:

$$\mathbf{Z}^{(2)}_{\text{DCT-II}} = \begin{pmatrix} & 1 & 1 & & \\ 1 & & & 1 & \\ 1 & & & & \ddots \\ & 1 & & & 1 \\ & & \ddots & & 1 \\ & & & 1 & 1 \end{pmatrix}, \quad \mathbf{Z}^{(3)}_{\text{DCT-II}} = \begin{pmatrix} & & 1 & 1 & & \\ & 1 & & & \ddots & \\ 1 & & & & & 1 \\ 1 & & & & & 1 \\ & \ddots & & & 1 & \\ & & 1 & 1 & & \end{pmatrix} \qquad (17)$$

For $\ell = N$, the derivations in (16) are also valid, but with $\mathbf{Z}^{(N)}_{\text{DCT-II}} = 2\mathbf{J}$. The rectangular patterns we observe in (17) can be simply extended to any arbitrary transform length N (e.g., all such operators with $N = 6$ are shown in Fig. 3(b)). We also show the associated eigenvalues of $\mathbf{Z}^{(\ell)}_{\text{DCT-II}}$ with arbitrary N in Table I. Note that all the operators and their associated graphs are sparse. In particular, each operator has at most $2N$ non-zero entries and its corresponding graph has at most $N-1$ edges.

B. Example–Length 4 DCT-II Operators

We show in Fig. 2(a) all sparse operators $\mathbf{Z}^{(\ell)}_{\text{DCT-II}}$ of DCT-II for $N = 4$. In fact, those matrices can be regarded as standard operators on different graphs: by defining $\mathbf{L}^{(\ell)}_{\text{DCT-II}} = 2\mathbf{I} - \mathbf{Z}^{(\ell)}_{\text{DCT-II}}$, we can view $\mathbf{L}^{(\ell)}_{\text{DCT-II}}$ as a Laplacian matrix of a different graph $\mathcal{G}^{(\ell)}_{\text{DCT-II}}$. For example, all the resulting $\mathbf{L}^{(\ell)}_{\text{DCT-II}}$'s and $\mathcal{G}^{(\ell)}_{\text{DCT-II}}$'s for a length-4 DCT-II are shown in Figure 2(b) and (c), respectively. The rectangular patterns we observe in (17) can be simply extended to any arbitrary transform length N. We also show the associated eigenvalues of $\mathbf{Z}^{(\ell)}_{\text{DCT-II}}$ with arbitrary N in Table I.

We observe that, among all graphs in Fig. 2(c), $\mathcal{G}^{(4)}_{\text{DCT-II}}$ is a disconnected graph with two connected components. It is associated to the operator

$$\mathbf{Z}^{(4)}_{\text{DCT-II}} = \mathbf{\Phi}_{\text{DCT-II}} \cdot \text{diag}(2, -2, 2, -2) \cdot \mathbf{\Phi}^{\top}_{\text{DCT-II}}.$$

Note that, while $\mathbf{Z}^{(4)}_{\text{DCT-II}}$ is associated to a disconnected graph, it can still be used as a graph operator for DCT-II filter because it is diagonalized by $\mathbf{\Phi}_{\text{DCT-II}}$. However, $\mathbf{Z}^{(4)}_{\text{DCT-II}}$, as well as its polynomials, have eigenvalues with multiplicity 2. This means that a filter whose frequency response has distinct values (e.g. low-pass filter with $h(\lambda_1) > \cdots > h(\lambda_4)$ cannot be realized as a PGF of $\mathbf{Z}^{(4)}_{\text{DCT-II}}$).

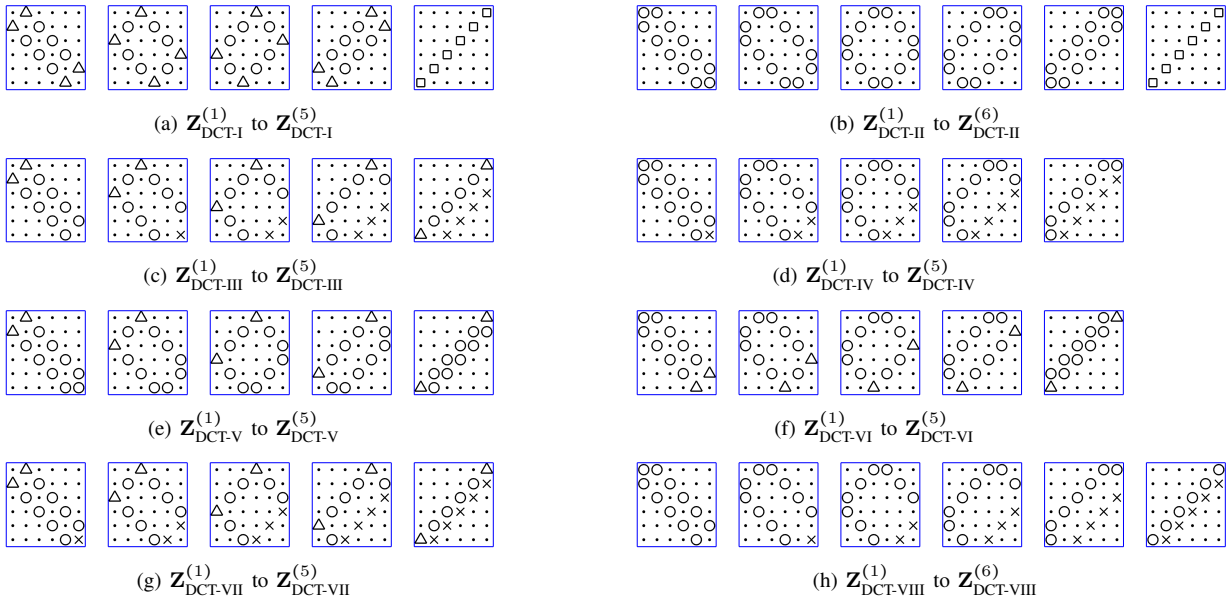
Based on the previous observation, we can see that those operators associated to disconnected graphs, and those having eigenvalues with high multiplicities lead to fewer degrees of freedoms in PGF and MPGF filter designs, as compared to an operators with distinct eigenvalues such as $\mathbf{Z}^{(1)}_{\text{DCT-II}}$.

C. Sparse Operators of 16 DTTs

The approach in Sec. III-A can be adapted to all 16 DTTs, so that their corresponding sparse operators can be obtained. In Table II, we show left and right boundary conditions of the DTTs. Those properties arise from even and odd symmetries of the cosine and sine functions [27], and can be easily verified based on DTT definitions in Table I. As an illustration, we present in Appendix A the derivations for DST-VI, DST-VII, and DCT-V, which share the same right boundary condition with DCT-II, but have different left boundary condition. Results for those DTTs with other combinations of left/right boundary condition can be easily extended.

Sparse operators and their associated eigenpairs for all DTTs are listed in Table I. Figs. 3 and 4 show the operators for $N = 6$, which can be easily extended to any arbitrary length. Interestingly, we observe that the non-zero entries in all sparse operators have rectangle-like patterns. Indeed, the 16 DTTs are constructed with combinations of 4 types of left boundary conditions and 4 types of right boundary conditions, associated to 4 types of upper-left rectangle edges and 4 types of lower-right rectangle edges in Figs. 3 and 4, respectively. For each sparse operator associated to any of the 16 DTTs, the associated graph Laplacian is given by the relationship as in (12): $\mathbf{L} = 2\mathbf{I} - \mathbf{Z}$. Then, as in the example in Fig. 2, the corresponding sparse graph can be obtained.

We also note that some of the sparse operators in Figs. 3 and 4 were already known. Those include $\mathbf{Z}^{(1)}_{\text{DCT-I}}$ [45], $\mathbf{I} + \mathbf{Z}^{(1)}_{\text{DCT-III}}$ and $\mathbf{I} + \mathbf{Z}^{(1)}_{\text{DCT-IV}}$ [46] (and [47] under a more general framework). In [27], left and right boundary conditions have been exploited to obtain sparse matrices



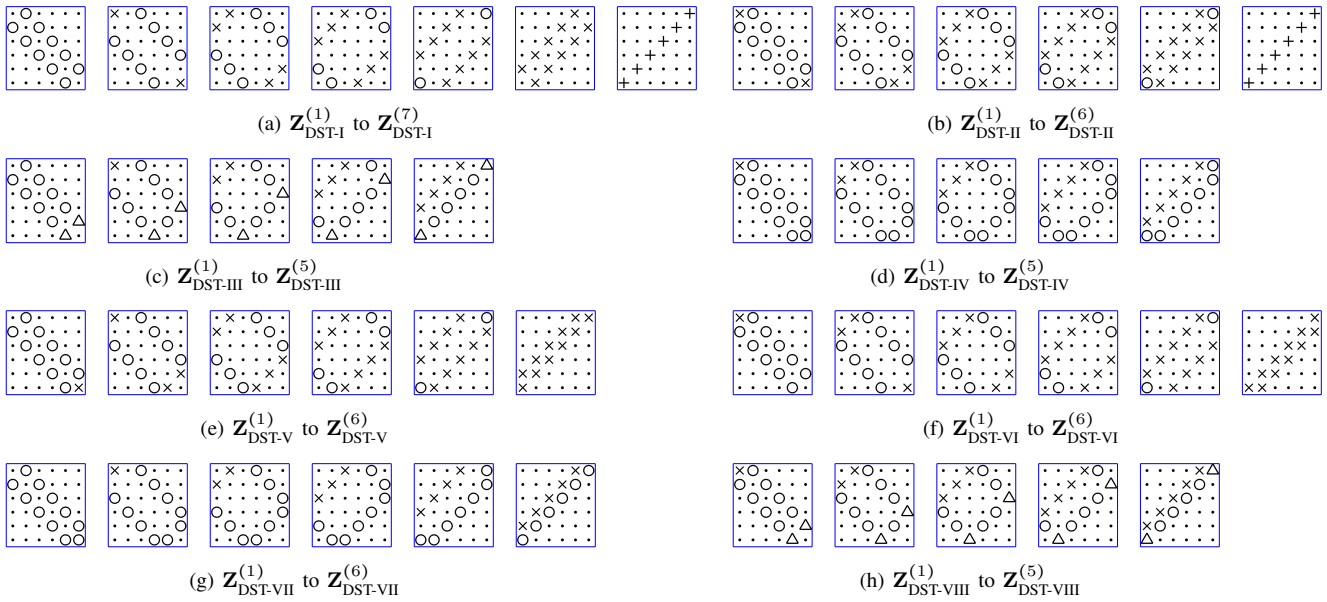
(a) $\mathbf{Z}^{(1)}_{\text{DCT-I}}$ to $\mathbf{Z}^{(5)}_{\text{DCT-I}}$ (b) $\mathbf{Z}^{(1)}_{\text{DCT-II}}$ to $\mathbf{Z}^{(6)}_{\text{DCT-II}}$

(c) $\mathbf{Z}^{(1)}_{\text{DCT-III}}$ to $\mathbf{Z}^{(5)}_{\text{DCT-III}}$ (d) $\mathbf{Z}^{(1)}_{\text{DCT-IV}}$ to $\mathbf{Z}^{(5)}_{\text{DCT-IV}}$

(e) $\mathbf{Z}^{(1)}_{\text{DCT-V}}$ to $\mathbf{Z}^{(5)}_{\text{DCT-V}}$ (f) $\mathbf{Z}^{(1)}_{\text{DCT-VI}}$ to $\mathbf{Z}^{(5)}_{\text{DCT-VI}}$

(g) $\mathbf{Z}^{(1)}_{\text{DCT-VII}}$ to $\mathbf{Z}^{(5)}_{\text{DCT-VII}}$ (h) $\mathbf{Z}^{(1)}_{\text{DCT-VIII}}$ to $\mathbf{Z}^{(6)}_{\text{DCT-VIII}}$

Fig. 3. Sparse graph operators with length $N = 6$ that associated to DCT-I to DCT-VIII. Different symbols represent different values: $\times = -1$, $\cdot = 0$, $\bigcirc = 1$, $\triangle = \sqrt{2}$, and $\square = 2$. The Laplacian associated to each of the operators is given by the relation: $\mathbf{L} = 2\mathbf{I} - \mathbf{Z}$.



(a) $\mathbf{Z}^{(1)}_{\text{DST-I}}$ to $\mathbf{Z}^{(7)}_{\text{DST-I}}$ (b) $\mathbf{Z}^{(1)}_{\text{DST-II}}$ to $\mathbf{Z}^{(6)}_{\text{DST-II}}$

(c) $\mathbf{Z}^{(1)}_{\text{DST-III}}$ to $\mathbf{Z}^{(5)}_{\text{DST-III}}$ (d) $\mathbf{Z}^{(1)}_{\text{DST-IV}}$ to $\mathbf{Z}^{(5)}_{\text{DST-IV}}$

(e) $\mathbf{Z}^{(1)}_{\text{DST-V}}$ to $\mathbf{Z}^{(6)}_{\text{DST-V}}$ (f) $\mathbf{Z}^{(1)}_{\text{DST-VI}}$ to $\mathbf{Z}^{(6)}_{\text{DST-VI}}$

(g) $\mathbf{Z}^{(1)}_{\text{DST-VII}}$ to $\mathbf{Z}^{(6)}_{\text{DST-VII}}$ (h) $\mathbf{Z}^{(1)}_{\text{DST-VIII}}$ to $\mathbf{Z}^{(5)}_{\text{DST-VIII}}$

Fig. 4. Sparse graph operators with length $N = 6$ that associated to DST-I to DST-VIII. Different symbols represent different values: $+ = -2$, $\times = -1$, $\cdot = 0$, $\bigcirc = 1$, and $\triangle = \sqrt{2}$. The Laplacian associated to each of the operators is given by the relation: $\mathbf{L} = 2\mathbf{I} - \mathbf{Z}$.

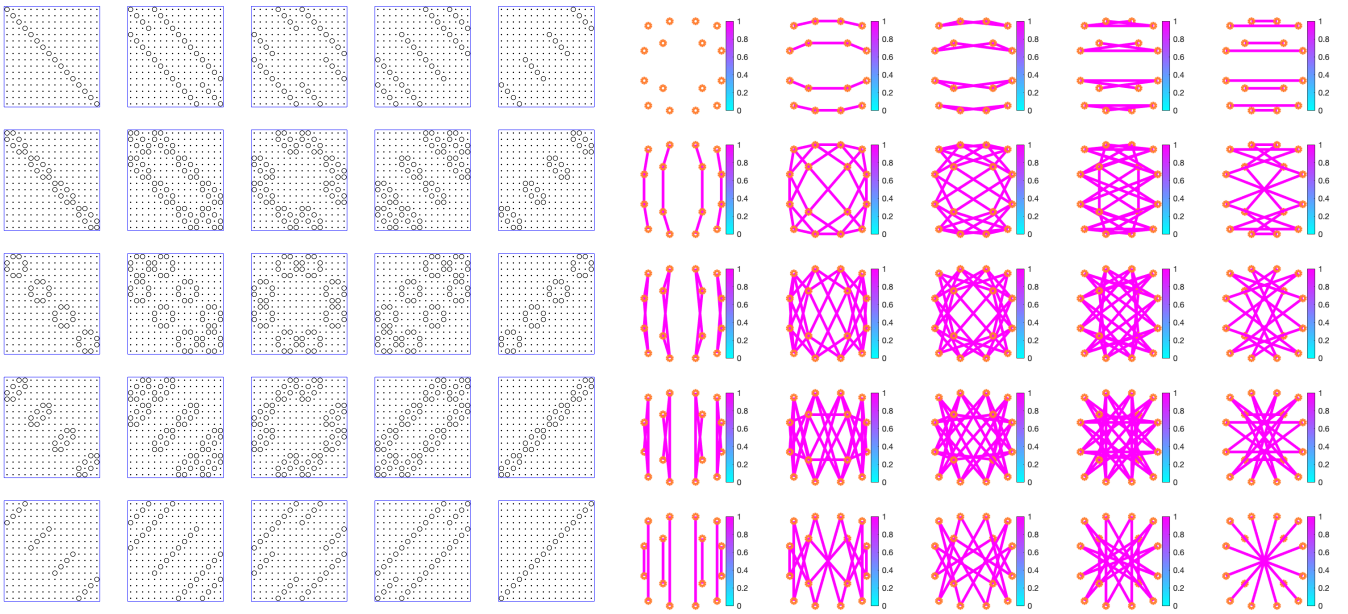
TABLE II
LEFT AND RIGHT BOUNDARY CONDITIONS (B.C.) OF 16 DTTS.

		Right boundary condition			
		$\phi_j(N+k) = \phi_j(N-k)$	$\phi_j(N+k) = -\phi_j(N-k)$	$\phi_j(N+k) = \phi_j(N-k+1)$	$\phi_j(N+k) = -\phi_j(N-k+1)$
Left b.c.	$\phi_j(k) = \phi_j(-k+2)$	DCT-I	DCT-III	DCT-V	DCT-VII
	$\phi_j(k) = -\phi_j(-k)$	DST-III	DST-I	DST-VII	DST-V
	$\phi_j(k) = \phi_j(-k+1)$	DCT-VI	DCT-VIII	DCT-II	DCT-IV
	$\phi_j(k) = -\phi_j(-k+1)$	DST-VIII	DST-VI	DST-IV	DST-II

with DTT eigenvectors, which correspond to the first operator $\mathbf{Z}^{(1)}$ for each DTT. However, to the best of our knowledge, graph operators with $\ell > 1$ (i.e., $\mathbf{Z}^{(2)}$ to $\mathbf{Z}^{(N-1)}$ for each DTT) have not been studied in the literature and are introduced here for the first time.

D. Sparse 2D DTT Operators

In image and video coding, the DTTs are often applied to 2D pixel blocks, where a combination of 1D DTTs can be applied to columns and rows of the blocks. We consider a $N_1 \times N_2$ block (with N_1 pixel



(a) (b)

Fig. 5. (a) Sparse operators and (b) graphs associated to 2D 4×4 DCT. Symbols $\cdot$ and $\circ$ in (a) represent 0 and 1, respectively. For visualization, coordinates in (b) are slightly shifted to prevent some edges from overlapping. Self-loops are not shown in the graphs. The graph in the top-left corner of (b) is associated to the identity matrix, whose corresponding graph contains self-loops only.

rows and N_2 pixel columns),

$$\begin{bmatrix} X_{1,1} & X_{1,2} & \dots & X_{1,N_2} \\ X_{2,1} & X_{2,2} & \dots & X_{2,N_2} \\ \vdots & \vdots & \vdots & \vdots \\ X_{N_1,1} & X_{N_1,2} & \dots & X_{N_1,N_2} \end{bmatrix}.$$

We use a 1D vector $\mathbf{x} \in \mathbb{R}^{N_1 N_2}$ to denote $\mathbf{X}$ with column-first ordering:

$$\mathbf{x} = (X_{1,1}, X_{2,1}, \dots, X_{N_1,1}, X_{1,2}, X_{2,2}, \dots, X_{N1,2}, \dots, X_{N_1,N_2})^\top$$

We assume that the GFT $\mathbf{\Phi} = \mathbf{\Phi}_r \otimes \mathbf{\Phi}_c$ is separable with row transform $\mathbf{\Phi}_r$ and column transform $\mathbf{\Phi}_c$. In such cases, sparse operators of 2D separable GFTs can be obtained from those of 1D transforms:

Proposition 3 (Sparse operators constructed by Kronecker products)**.** *Let $\mathbf{\Phi} = \mathbf{\Phi}_r \otimes \mathbf{\Phi}_c$ with $\mathbf{\Phi}_r$ and $\mathbf{\Phi}_c$ being orthogonal transforms, and let $\mathcal{Z}_r$ and $\mathcal{Z}_c$ be the set of sparse operators associated to $\mathbf{\Phi}_r$ and $\mathbf{\Phi}_c$, respectively. Denote the eigenpairs associated to the operators of $\mathcal{Z}_r$ and $\mathcal{Z}_c$ as $(\lambda_{r,j}, \boldsymbol{\phi}_{r,j})$ and $(\lambda_{c,k}, \boldsymbol{\phi}_{c,k})$ with $j = 1, \dots, N_1$ and $k = 1, \dots, N_2$. Then,*

$$\mathcal{Z} = \{\mathbf{Z}_r \otimes \mathbf{Z}_c,\ \mathbf{Z}_r \in \mathcal{Z}_r,\ \mathbf{Z}_c \in \mathcal{Z}_c\}$$

is a set of sparse operators corresponding to $\mathbf{\Phi}_r \otimes \mathbf{\Phi}_c$, with associated eigenpairs $(\lambda_{r,j}\lambda_{c,k}, \boldsymbol{\phi}_{r,j} \otimes \boldsymbol{\phi}_{c,k})$.

Proof: Let $\mathbf{Z}_r^{(1)}, \dots, \mathbf{Z}_r^{(M_1)}$ be sparse operators in $\mathcal{Z}_r$ with associated eigenvalues contained in vectors $\boldsymbol{\lambda}_r^{(1)}, \dots, \boldsymbol{\lambda}_r^{(M_1)}$, respectively. Also let $\mathbf{Z}_c^{(1)}, \dots, \mathbf{Z}_c^{(M_1)}$ be those in $\mathcal{Z}_c$ with eigenvalues in $\boldsymbol{\lambda}_c^{(1)}, \dots, \boldsymbol{\lambda}_c^{(M_2)}$, respectively. We note that

$$\mathbf{Z}_r^{(m_1)} = \mathbf{\Phi}_r \cdot \mathrm{diag}(\boldsymbol{\lambda}_r^{(m_1)}) \cdot \mathbf{\Phi}_r^\top, \quad m_1 = 1, \dots, M_1,$$
$$\mathbf{Z}_c^{(m_2)} = \mathbf{\Phi}_c \cdot \mathrm{diag}(\boldsymbol{\lambda}_c^{(m_2)}) \cdot \mathbf{\Phi}_c^\top, \quad m_2 = 1, \dots, M_2.$$

Applying a well-known Kronecker product identity [48], we obtain

$$\mathbf{Z}_r^{(m_1)} \otimes \mathbf{Z}_c^{(m_2)} = \mathbf{\Phi} \cdot \mathrm{diag}(\boldsymbol{\lambda}_r^{(m_1)} \otimes \boldsymbol{\lambda}_c^{(m_2)}) \cdot \mathbf{\Phi}^\top. \qquad \square$$

Proposition 3 allows us to obtain sparse operators for all 2D DTTs, where $\mathbf{\Phi}_c$ and $\mathbf{\Phi}_r$ are allowed to be the same. An example is shown in Fig. 5, where $\mathbf{\Phi}_c = \mathbf{\Phi}_r$ is the length-4 DCT-II, and $\mathbf{\Phi}$ is the 4×4 2D DCT.

IV. Sparse Operators for Transforms Beyond DTTs

In this section, we will discuss sparse operators beyond 16 types of DTTs. In particular, the operators associated with DFT will be shown in Sec. IV-A, and those corresponding to the Hadamard transform will be presented in Sec. IV-B. We will also make some remarks on the retrieval of operators for arbitrary GFTs in Sec.IV-C.

A. Sparse DFT Operators

Essentially, graph operators of the DFT are circular shift operators, and the vertex domain filter operation corresponds to the FIR filtering operation in classical discrete signal processing (DSP). Since DFT properties are well known, we briefly summarize the main results with some details omitted.

We consider a fundamental circulant matrix, which corresponds to a circular shift operator:

$$\mathbf{C} = \begin{pmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{pmatrix} = \frac{1}{N}\mathbf{U}_{\mathrm{DFT}}^H \begin{pmatrix} e^{-j\frac{2\pi \cdot 0}{N}} & & \\ & \ddots & \\ & & e^{-j\frac{2\pi(N-1)}{N}} \end{pmatrix} \mathbf{U}_{\mathrm{DFT}}.$$

Note that a real-valued matrix $\mathbf{M}$ is diagonalizable by the DFT matrix if and only if $\mathbf{M}$ is circulant [49]. Thus, $\mathcal{C} = \{\mathbf{I}, \mathbf{C}, \mathbf{C}^2, \dots, \mathbf{C}^{N-1}\}$ is a set of sparse DFT operators, and the DFT filter output can be expressed as a linear combination of circular shifted version of the input signal. This relationship is consistent with the classical convolution-multiplication property of the DFT. We also highlight that, since the DFT is complex-valued, it diagonalizes particular non-symmetric matrices such as $\mathbf{C}$. On the contrary, in the context of DTT filtering, we consider matrix diagonalization over the real numbers, so the DTT operators we have derived are all symmetric.

B. Sparse Operators for Hadamard Transforms

The Hadamard transform matrix of size $2^m \times 2^m$ is defined as

$$\mathbf{H}_0 := 1, \quad \mathbf{H}_m := \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{H}_{m-1} & \mathbf{H}_{m-1} \\ \mathbf{H}_{m-1} & -\mathbf{H}_{m-1} \end{pmatrix} \text{ for } m \geq 1.$$

Obviously, $\mathbf{H}_1 = \mathbf{U}_{\text{DCT-II}}$ with $N = 2$ and thus the set of its sparse operators can be written as $\mathcal{I}_1 = \{2\mathbf{I}, 2\mathbf{J}\}$. For $m > 1$, by definition, we can denote $\mathbf{H}_m$ as $\mathbf{H}_m = \mathbf{H}_1 \otimes \mathbf{H}_{m-1}$. Then, applying Proposition 3, we can also construct the set of operators of $\mathbf{H}_m$ recursively by

$$\mathcal{I}_m = \{\mathbf{Z}_1 \otimes \mathbf{Z}_2 \mid \mathbf{Z}_1 \in \mathcal{I}_1, \mathbf{Z}_2 \in \mathcal{I}_{m-1}\}.$$

Similarly, the associated eigenvalues can also be obtained based on those of DCT-II and the Kronecker products. It can be shown that all the operators generated will not be repeated, so $|\mathcal{I}_m| = 2^m$. In addition, they are all zero-one matrices, which has low computational complexity when multiplied to an arbitrary input signal and may potentially be useful for Hadamard domain filtering [50].

C. Remarks on Graph Operators of Arbitrary GFTs

Obtaining multiple sparse operators $\mathbf{Z}^{(k)}$ for an arbitrary fixed GFT $\mathbf{\Phi} \in \mathbb{R}^{N\times N}$ is a challenging problem in general. Start by noting that given a graph Laplacian associated to $\mathbf{\Phi}$ be $\mathbf{L}$, with λ_j the eigenvalue of $\mathbf{L}$ associated to eigenvector $\boldsymbol{\phi}_j$, if the graph does not have any self-loops, the Laplacian of the complement graph [51]

$$\mathbf{L}^c := N w_{\max} \mathbf{I} - w_{\max} \mathbf{1}\mathbf{1}^\top - \mathbf{L},$$

has eigenpairs $(0, \boldsymbol{\phi}_1)$ and $(n - \lambda_j, \boldsymbol{\phi}_j)$ for $j = 2, \ldots, N$. However, $\mathbf{L}^c$ will be a dense matrix when $\mathbf{L}$ is sparse, and thus may not be suitable for an efficient MPGF design. We next summarize some additional results on the retrieval of sparse graph operators, with more details given in Appendix B.

1) Characterization of Sparse Laplacians of a Common GFT: Extending a key result in [52], we can characterize the set of all graph Laplacians (i.e., those that satisfy (4) with non-negative edge and self-loop weights) sharing a given GFT $\mathbf{\Phi}$ by a convex polyhedral cone. In particular, those graph Laplacians that are the most sparse among all would correspond to the edges of a polyhedral cone (i.e., where the faces of the cone meet each other). However, the enumeration of edges is in general an NP-hard problem since the number of polyhedron vertices or edges can be a combinatorial number of N.

2) Construction of Sparse Operators from Symmetric Graphs: If a graph with Laplacian $\mathbf{L}$ satisfies the symmetry property defined in [11], then we can construct a sparse operator in addition to $\mathbf{L}$. In particular, we first characterize a node pairing function by an involution $\varphi : \mathcal{V} \to \mathcal{V}$, which is a permutation whose inverse is itself (i.e., φ satisfies $\varphi(\varphi(i)) = i$ for all $i \in \mathcal{V}$). In this way, we call a graph φ-symmetric if $w_{i,j} = w_{\varphi(i),\varphi(j)}$ for all $i, j \in \mathcal{V}$. For such a graph, a sparse operator can be constructed as follows:

Lemma 1. *Given a φ-symmetric graph $\mathcal{G}$ with Laplacian $\mathbf{L}$, we can construct a graph $\overline{\mathcal{G}_\varphi}$ by connecting nodes i and j with edge weight 1 for all node pairs (i, j) with $\varphi(i) = j, i \neq j$). In this way, the Laplacian $\overline{\mathbf{L}_\varphi}$ of $\overline{\mathcal{G}_\varphi}$ commutes with $\mathbf{L}$.*

The proof is presented in Appendix C.

V. Graph Filter Design with Sparse Operators

In this section, we introduce some filter design approaches based on sparse operators for DTTs. We start by summarizing the least squares design method in Sec. V-A. We also propose a minimax filter design in Sec. V-B for both PGF and MPGF.

A. Least Squares (LS) Graph Filter

For an arbitrary graph filter $\mathbf{H}^*$, its frequency response, $\mathbf{h}^* = (h^*(\lambda_1), \ldots, h^*(\lambda_N))^\top$, can be approximated with a filter $\mathbf{H}_{\mathcal{Z},K}$ in (3) by designing a set of coefficients $\mathbf{g}$ as in (10) or (11). Let $h(\lambda_j)$ be the frequency response corresponding to $\mathbf{H}_{\mathcal{Z},K}$, then one way to obtain $\mathbf{g}$ is through a least squares solution:

$$\begin{aligned} \mathbf{g}^* &= \operatorname*{argmin}_{\mathbf{g}} \quad \sum_{j=1}^{N} \left(h^*(\lambda_j) - h(\lambda_j)\right)^2 \\ &= \operatorname*{argmin}_{\mathbf{g}} \quad \sum_{j=1}^{N} \left(h^*(\lambda_j) - p_K(\lambda_j^{(1)}, \ldots, \lambda_j^{(M)})\right)^2 \\ &= \operatorname*{argmin}_{\mathbf{g}} \quad \left\|\mathbf{h}^* - \mathbf{\Pi}_K(\boldsymbol{\lambda}^{(1)}, \ldots, \boldsymbol{\lambda}^{(M)}) \cdot \mathbf{g}\right\|^2, \end{aligned} \tag{18}$$

where $\mathbf{\Pi}_K$ for $K = 1$ and $K = 2$ are shown in (19).

This formulation can be generalized to a weighted least squares problem, where we allow different weights for different graph frequencies. This enables us to approximate the filter in particular frequencies with higher accuracy. In this case, we consider

$$\begin{aligned} \mathbf{g}^* &= \operatorname*{argmin}_{\mathbf{g}} \quad \sum_{j=1}^{N} \rho_i^2 \left(h^*(\lambda_j) - h(\lambda_j)\right)^2 \\ &= \operatorname*{argmin}_{\mathbf{g}} \quad \left\|\text{diag}(\boldsymbol{\rho})(\mathbf{h}^* - \mathbf{\Pi}_K \cdot \mathbf{g})\right\|^2, \end{aligned} \tag{20}$$

where $\rho_i \geq 0$ is the weight corresponding to λ_i. Note that when $\boldsymbol{\rho} = \mathbf{1}$, the problem (20) reduces to (18).

When $\mathbf{g}$ is sparser, (i.e., its ℓ_0 norm is smaller), fewer terms will be involved in the polynomial p_K, leading to a lower complexity for the filtering operation. This ℓ_0-constrained problem can be viewed as a sparse representation of $\text{diag}(\boldsymbol{\rho})\mathbf{h}^*$ in an overcomplete dictionary $\text{diag}(\boldsymbol{\rho})\mathbf{\Pi}_K$. Well-known methods for this problem include the orthogonal matching pursuit (OMP) algorithm [53], and the optimization with a sparsity-promoting ℓ_1 constraint:

$$\operatorname*{minimize}_{\mathbf{g}} \quad \left\|\text{diag}(\boldsymbol{\rho})(\mathbf{h}^* - \mathbf{\Pi}_K \cdot \mathbf{g})\right\|^2 \quad \text{subject to} \quad \|\mathbf{g}\|_1 \leq \tau, \tag{21}$$

where τ is a pre-chosen threshold. In fact, this formulation can be viewed as an extension of its PGF counterpart [18] to an MPGF setting. Note that (21) is a ℓ_1-constrained least squares problem (a.k.a., the LASSO problem), where efficient solvers are available [54].

Compared to conventional PGF $\mathbf{H}$ in (2), the implementation with $\mathbf{H}_{\mathcal{Z},K}$ has several advantages. First, when $K = 1$, the MPGF (10) is a linear combination of different sparse operators, which is amenable to parallelization. This is in contrast to high degree PGFs based on (2), which require applying the graph operator repeatedly. Second, $\mathbf{H}_{\mathcal{Z},K}$ is a generalization of $\mathbf{H}$ and provides more degrees of freedom, which provides more accurate approximation with equal or lower order polynomial. Note that, while the eigenvalues of $\mathbf{Z}^k$ for $k = 1, 2, \ldots$ are typically all increasing (if $\mathbf{Z} = \mathbf{L}$) or decreasing (if, for instance, $\mathbf{Z} = 2\mathbf{I} - \mathbf{L}$), those of different $\mathbf{Z}^{(m)}$'s have more diverse

$$\mathbf{\Pi}_1(\boldsymbol{\lambda}^{(1)}, \ldots, \boldsymbol{\lambda}^{(M)}) = \begin{pmatrix} 1 & \lambda_1^{(1)} & \cdots & \lambda_1^{(M)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \lambda_N^{(1)} & \cdots & \lambda_N^{(M)} \end{pmatrix}, \quad \mathbf{\Pi}_2(\boldsymbol{\lambda}^{(1)}, \ldots, \boldsymbol{\lambda}^{(M)}) = \begin{pmatrix} 1 & \lambda_1^{(1)} & \cdots & \lambda_1^{(M)} & \lambda_1^{(1)}\lambda_1^{(1)} & \lambda_1^{(1)}\lambda_1^{(2)} & \cdots & \lambda_1^{(M)}\lambda_1^{(M)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ 1 & \lambda_N^{(1)} & \cdots & \lambda_N^{(M)} & \lambda_N^{(1)}\lambda_N^{(1)} & \lambda_N^{(1)}\lambda_N^{(2)} & \cdots & \lambda_N^{(M)}\lambda_N^{(M)} \end{pmatrix}. \tag{19}$$

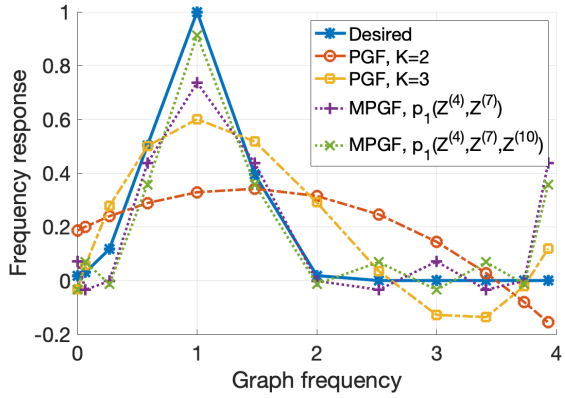


Fig. 6. An example for PGF and MPGF fitting results on a length 12 line graph. The desired frequency response is $h^*(\lambda) = \exp(-4(\lambda-1)^2)$. The PGF and MPGF filters have been optimized based on (2) and (18).

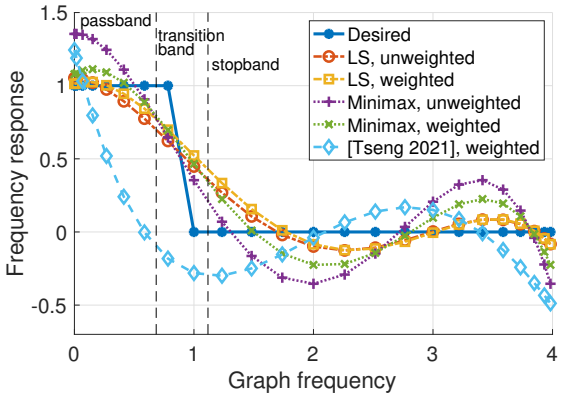


Fig. 7. Example illustrating the frequency responses of degree $K = 4$ PGF with least squares (LS) and minimax criteria, with weighted or unweighted settings. The filters are defined on a length 24 line graph. In the weighted setting, weights ρ_i are chosen to be 2, 0, and 1 for passband, transition band, and stopband, respectively.

distributions (i.e., increasing, decreasing, or non-monotonic). Thus, MPGFs provide better approximations for filters with non-monotonic frequency responses. For example, we demonstrate in Fig. 6 the resulting PGF and MPGF for a bandpass filter. We can see that, for $K = 2$ and $K = 3$, a degree-1 MPGF with K operators gives a higher approximation accuracy than a degree-K PGF, while they have a similar complexity.

B. Minimax Graph Filter

The minimax approach is a popular filter design method in classical signal processing. The goal is to design an length-K FIR filter whose frequency response $G(e^{j\omega})$ approximates the desired frequency response $H(e^{j\omega})$ in a way that the maximum error within some range of frequency is minimized. A standard design method is the Parks-McClellan algorithm, which is a variation of the Remez exchange algorithm [55].

Here, we explore minimax design criteria for graph filters. We denote $h^*(\lambda)$ the desired frequency response, and $g(\lambda)$ the polynomial filter that approximates $h^*(\lambda)$. Source code for the proposed minimax graph filter design methods can be found in [56].

1) Polynomial Graph Filter: Let $g(\lambda)$ be the PGF with degree K given by (2). Since graph frequencies $\lambda_1, \ldots, \lambda_N$ are discrete, we only need to minimize the maximum error between h^* and g at frequencies $\lambda_1, \ldots, \lambda_N$. In particular, we would like to solve polynomial coefficients g_i:

$$\underset{\mathbf{g}}{\text{minimize}} \quad \underbrace{\max_i \rho_i \left| h^*(\lambda_i) - \sum_{j=0}^{K} g_j \lambda_i^j \right|}_{\|\text{diag}(\boldsymbol{\rho})\left(\mathbf{h}^* - \boldsymbol{\Psi}\mathbf{g}\right)\|_\infty}$$

where $\boldsymbol{\Psi}$ is the matrix in (6), ρ_i is the weight associated to λ_i and $\|\cdot\|_\infty$ represents the infinity norm. Note that, when $K \geq N-1$ and $\boldsymbol{\Psi}$ is full row rank, then $\mathbf{h}^* = \boldsymbol{\Psi}\mathbf{g}$ can be achieved with $\mathbf{g} = \boldsymbol{\Psi}^\dagger \mathbf{h}^*$. Otherwise, we reduce this problem by setting $\epsilon = \|\text{diag}(\boldsymbol{\rho})\left(\mathbf{h}^* - \boldsymbol{\Psi}\mathbf{g}\right)\|_\infty$:

$$\underset{\mathbf{g},\,\epsilon}{\text{minimize}} \quad \epsilon \quad \text{subject to} \quad -\epsilon\mathbf{1} \leq \text{diag}(\boldsymbol{\rho})\left(\mathbf{h}^* - \boldsymbol{\Psi}\mathbf{g}\right) \leq \epsilon\mathbf{1}, \quad (22)$$

whose solution can be efficiently obtained with a linear programming solver.

2) Multivariate Polynomial Graph Filter: Now we consider $g(\lambda)$ a graph filter with M graph operators with degree K, as in (3). In this case, we can simply extend the problem (22) to

$$\underset{\mathbf{g},\,\epsilon}{\text{minimize}} \quad \epsilon \quad \text{subject to} \quad -\epsilon\mathbf{1} \leq \text{diag}(\boldsymbol{\rho})\left(\mathbf{h}^* - \boldsymbol{\Pi}_K\mathbf{g}\right) \leq \epsilon\mathbf{1}, \quad (23)$$

	PGF	MPGF
Least squares	$\min_{\mathbf{g}} \quad \|\text{diag}(\boldsymbol{\rho})(\mathbf{h} - \boldsymbol{\Psi}\mathbf{g})\|^2$	(20)
Minimax	(22)	(23)

TABLE III
LEAST SQUARES AND MINIMAX DESIGN APPROACHES OF FOR PGF AND MPGF, WITH WEIGHTS ρ_i ON DIFFERENT GRAPH FREQUENCIES.

where a ℓ_1 or ℓ_0 norm constraint on $\mathbf{g}$ can also be considered.

To summarize, we show in Table III the objective functions of least squares and minimax designs with PGF and MPGF, where weights on different graph frequencies are considered. Note that the least squares PGF design shown in Table III is a simple extension of the unweighted design (6) in [15].

Using an ideal low-pass filter as the desired filter, we show a toy example with degree-4 PGF in Fig. 7. When different weights ρ_i are used for passband, transition band, and stopband, approximation accuracies differ for different graph frequencies. By comparing LS and minimax results in a weighted setting, we also see that the minimax criterion yields a smaller maximum error within the passband (see the last frequency bin in passband) and stopband (see the first frequency bin in stopband).

We also highlight that our minimax criterion is different from that proposed in [35]. While we consider the discrete infinity norm, $\max_i |h^*(\lambda_i) - h(\lambda)|$ on the graph frequency domain, [35] approaches the continuous infinity norm $\|H(\omega) - D(\omega)\|_\infty$ on the classical frequency domain $\omega \in [0, \pi]$ through the bijective mapping $\omega = \cos^{-1}(1-\lambda)$, with $\lambda \in [0, 2]$ the normalized graph frequency. Fig. 7 shows the ideal low pass filter designed with the approach in [35] (using the same weights for fair comparison). We can see that its frequency response is different from that obtained with our approach and this design leads to a larger $\max_i |h^*(\lambda_i) - h(\lambda)|$ error. As for computational complexity, the approach in [35] is based on the Park-McClellan algorithm, which requires solving a linear system and finding local maxima of a continuous function for each iteration. On the other hand, our algorithm only requires solving a linear program, which has a much lower complexity.

VI. Graph Filter Implementation with Sparse Operators

In this section, we will give details of distributed implementation of PGF and MPGFs in Sec. VI-A. Then, in Sec. VI-B we show that

Algorithm 1 Distributed Implementation with Graph Operator $\mathbf{Z}$

Require: Outputs at node i: $(\mathbf{Zt})_i$
Ensure: Inputs at node i: t_i, $\mathbf{Z}$
1: Transmit t_i to all neighbors $\mathcal{N}_i := \{j : z_{ij} \neq 0\}$
2: Receive t_j from all neighbors $j \in \mathcal{N}_i$
3: Obtain output value $(\mathbf{Zt})_i = \sum_{j\in\mathcal{N}} z_{ij} t_j$

Algorithm 2 Distributed MPGF Implementation

Require: Output $\mathbf{y} \in \mathbb{R}^N$
Ensure: Input $\mathbf{x} \in \mathbb{R}^N$, and MPGF $\mathbf{H} = \sum_{r=1}^{R} \mathbf{H}_m$, with $\mathbf{H}_r = \sigma_r \prod_{k=1}^{K_r} \mathbf{Z}^{(i_{r,k})}$, $\sigma_r \in \mathbb{R}$, $i_{r,k} \in \{1, \ldots, M\}$, and $K_r \in \mathbb{N}$
1: $\mathbf{y} \leftarrow \mathbf{0}$
2: **for** $r = 1, \ldots, R$ **do**
3: $\quad \mathbf{t} \leftarrow \sigma_r \mathbf{x}$
4: $\quad$ **for** $k = 1, \ldots, K_r$ **do**
5: $\quad\quad \mathbf{t} \leftarrow \mathbf{Z}^{(i_{r,k})}\mathbf{t}$ // using Algorithm 1
6: $\quad$ **end for**
7: $\quad \mathbf{y} \leftarrow \mathbf{y} + \mathbf{t}$
8: **end for**

weighted energy in graph frequency domain can also be efficiently approximated using multiple graph operators. Complexity analysis will be presented in Sec. VI-C.

A. Distributed Graph Filter Implementation

As introduced in Sec. II-B, the output $\mathbf{y} = \mathbf{Hx}$ with input signal $\mathbf{x}$ and a PGF $\mathbf{H} = \sum_{j=0}^{K} g_j \mathbf{Z}^j$ can be obtained with a iterative procedure based on the well-known Horner scheme:

1) $\mathbf{t}^{(0)} = g_K \mathbf{x}$
2) **for** $j = 1, \ldots, K$ **do** $\mathbf{t}^{(j)} = \mathbf{Z}\mathbf{t}^{(j-1)} + g_{K-j}\mathbf{x}$
3) $\mathbf{y} = \mathbf{t}^{(K)}$,

which is led by the nested expression of polynomial:

$$h(x) := \sum_{j=0}^{K} g_j x^j = g_0 + x\left(g_1 + x\left(g_2 + \cdots + x\left(g_{K-1} + x\, g_K\right)\right)\right) \quad (24)$$

Note that a PGF can be performed in a distributed fashion. Let $\mathbf{x} \in \mathbb{R}^N$ be a signal in a network with N nodes. Then, for the iterative step, since $(\mathbf{Zt})_i = \sum_j z_{ij} t_i$, the task for node i is to exchange messages with its neighbors (characterized by non-zero entries in the i-th row of $\mathbf{Z}$). The detailed procedure for this operation is presented in Algorithm 1. With sparse DTT operators,

Similarly, MPGFs can also be implemented in a distributed manner using Algorithm 1 as a building block. First, we express an MPGF as a sum of monomials $\mathbf{H} = \sum_r \mathbf{H}_r$, where $\mathbf{H}_r = \sigma_r \prod_{k=1}^{K_r} \mathbf{Z}^{(i_{r,k})}$ with total degree $K_r \in \mathbb{N}$, coefficient $\sigma_r \in \mathbb{R}$, and operators $\mathbf{Z}^{(i_{r,k})}$ with $i_{r,k} \in \{1, \ldots, M\}$. Second, we multiply $\mathbf{x}$ by each monomial via $\mathbf{y}_r = \mathbf{H}_r \mathbf{x}$ by sequentially applying the operators $\mathbf{Z}^{(i_{r,k})}$ to $\mathbf{x}$, with the distributed implementation in Algorithm 1. Third, we obtain $\mathbf{y}$ by taking the sum of all $\mathbf{y}_r$'s. The detailed procedures are presented in Algorithm 2. We also note that, nested expression such as (24) would also allow us to reuse intermediate results for faster MPGF evaluation. However, finding the nested expression that leads to fastest evaluation of an arbitrary multivariate polynomial is a challenging problem, where reasonable solutions can be found by greedy method such as [57].

B. Weighted GFT Domain Energy Evaluation

Let $\mathbf{x}$ be a signal and $\mathbf{\Phi}$ be a GFT to be applied, we consider a weighted sum of squared GFT coefficients:

$$\mathcal{C}_{\mathbf{\Phi}}(\mathbf{x}; \mathbf{q}) = \sum_{i=1}^{N} q_i (\boldsymbol{\phi}_i^\top \mathbf{x})^2, \quad (25)$$

where arbitrary weights $\mathbf{q} = (q_1, \ldots, q_N)^\top$ can be considered. Then $\mathcal{C}_{\mathbf{\Phi}}(\mathbf{x}; \mathbf{q})$ has a similar form to the Laplacian quadratic form (5), since

$$\mathbf{x}^\top \mathbf{L} \mathbf{x} = \sum_{l=1}^{N} \lambda_l (\boldsymbol{\phi}_l^\top \mathbf{x})^2. \quad (26)$$

Note that computation of $\mathbf{x}^\top \mathbf{L} \mathbf{x}$ using (5) can be done in the vertex domain, and does not require the GFT coefficients. This provides a low complexity implementation than (26), especially when the graph is sparse (i.e., few edges and self-loops).

Similar to vertex domain Laplacian quadratic form computation (5), we note that $\mathcal{C}_{\mathbf{\Phi}}(\mathbf{x}; \mathbf{q})$ can also be realized as a quadratic form:

$$\mathcal{C}_{\mathbf{\Phi}}(\mathbf{x}; \mathbf{q}) = \sum_{i=1}^{N} q_i (\boldsymbol{\phi}_i^\top \mathbf{x})^2 = \mathbf{x}^\top \underbrace{\left(\mathbf{\Phi} \cdot \mathrm{diag}(\mathbf{q}) \cdot \mathbf{\Phi}^\top\right)}_{\mathbf{H}_\mathbf{q}} \mathbf{x}, \quad (27)$$

where $\mathbf{H}_\mathbf{q}$ can be viewed as a graph filter with frequency response $h_\mathbf{q}(\lambda_i) = q_i$. Thus, we can approximate $\mathbf{H}_\mathbf{q}$ with a sparse filter $\mathbf{H}_{\hat{\mathbf{q}}}$ such that $\mathbf{x}^\top \mathbf{H}_{\hat{\mathbf{q}}} \mathbf{x}$ approximates $\mathcal{C}_{\mathbf{\Phi}}(\mathbf{x}; \mathbf{q})$. For example, if we consider a polynomial with degree 1 as in (10), we have

$$\mathbf{x}^\top \underbrace{\left[g_0 \mathbf{I} + \sum_{m=1}^{M} g_m \mathbf{Z}^{(m)}\right]}_{\mathbf{H}_{\hat{\mathbf{q}}}} \mathbf{x} = \sum_{i=1}^{N} \underbrace{\left(g_0 + \sum_{m=1}^{M} g_m \lambda_i^{(m)}\right)}_{\hat{q}_i} (\boldsymbol{\phi}_i^\top \mathbf{x})^2. \quad (28)$$

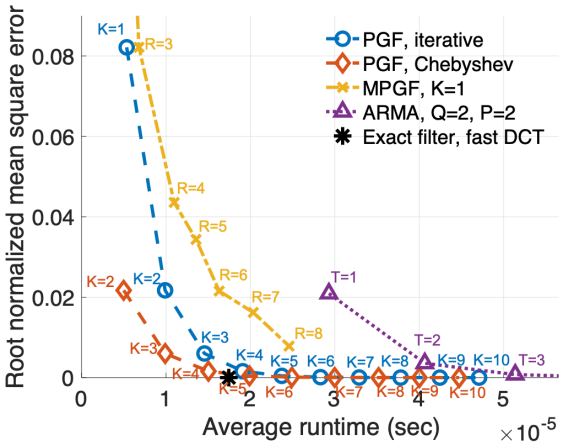
The left hand side can be computed efficiently if there are only a few nonzero g_m, making $\mathbf{H}_{\hat{\mathbf{q}}}$ sparse. The right hand side can be viewed as a proxy of (25) if g_m's are chosen such that $\hat{q}_i \approx q_i$. Such coefficients g_m can be obtained by solving (21) with $\mathbf{h}^* = \mathbf{q}$.

C. Complexity Analysis

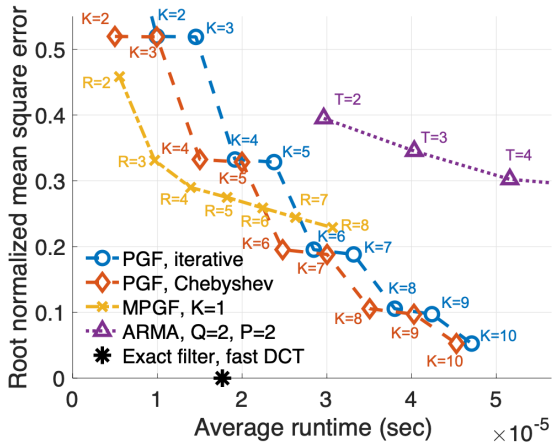
For a graph with N nodes and E edges, it has been shown in [13] that a degree-K PGF has $\mathcal{O}(KE)$ complexity. For an MPGF with R terms, we denote E' the maximum number of nonzero elements of the operator among all operators involved. Each term of MPGF requires at most $\mathcal{O}(KE')$ operations, so the overall complexity of an MPGF is $\mathcal{O}(KRE')$, assuming that the maximum total degree of this MPGF is K. We note that for DTT filters, the sparsity of all operators we have introduced is at most $2N$. Thus, complexities of PGF and MPGF can be reduced to $\mathcal{O}(KN)$ and $\mathcal{O}(KRN)$, respectively, which account for $\mathcal{O}(K)$ and $\mathcal{O}(KR)$ per node in distributed implementation. We note that $\mathcal{O}(KRN)$ is not a tight upper bound for the complexity if many terms of the MPGF have lower degrees than K. In addition, the polynomial degree required by an MPGF to reach a similar accuracy as a PGF can achieve may be lower. Thus, an MPGF does not necessarily have higher complexity than a PGF that bring a similar approximation accuracy. Indeed, MPGF implementation may be further optimized by parallelizing the computation associated to different graph operators.

VII. Experiments

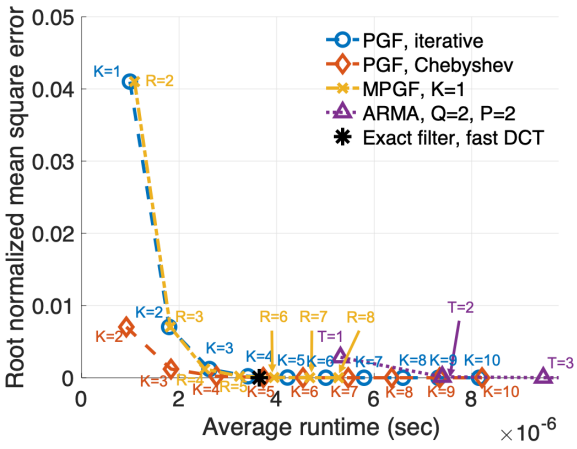
We consider two experiments to validate the filter design approaches. In Sec. VII-A, we evaluate the complexity of PGF and MPGF for DCT-II, and compare the trade-off between complexity and filter approximation accuracy as compared to conventional implementations in the DCT domain. In Sec. VII-B we implement DTT filters in a state-of-the-art video encoder–AV1, where we obtain a computational speedup in transform type search.



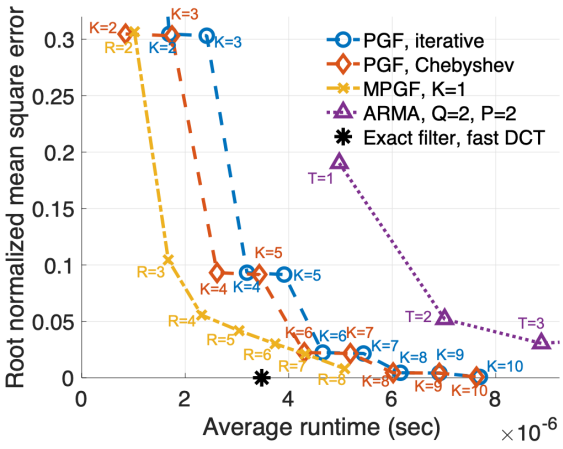
(a) Tikhonov, 16×16 grid



(b) Bandpass exponential, 16×16 grid



(c) Tikhonov, length-64 line graph



(d) Bandpass exponential, length-64 line graph

Fig. 8. Runtime vs approximation error for (a)(c) Tikhonov DCT filter, (b)(d) bandpass exponential DCT filter. Those filters are defined based on two different graphs: (a)(b) 16×16 grid, (c)(d) length-64 line graph. Different PGF degrees K, MPGF operators involved R, and ARMA iteration numbers T, are labelled in the figures.

A. Filter Approximation Accuracy with Respect to Complexity

In the first experiment, we implement several DCT filters that have been used in the literature. The graphs we use for this experiment include a 16×16 grid, and a length-64 line graph. Those filters are implemented in C in order to fairly evaluate computational complexity under an environment close to hardware[3].

1) Comparison among filter implementations: Here, the following filters are considered:

- *Tikhonov filter:* given $\mathbf{z} = \mathbf{x} + \mathbf{n}$, a noisy observation of signal $\mathbf{x}$, the denoising problem can be formulated as a regularized least squares problem:

$$\underset{\mathbf{x}}{\text{minimize}} \quad \|\mathbf{x} - \mathbf{z}\|^2 + \mu \mathbf{x}^\top \mathbf{L} \mathbf{x}.$$

  The solution is given by $\hat{\mathbf{x}} = \mathbf{H}_t \mathbf{x}$, where $\mathbf{H}_t = (\mathbf{I} + \mu \mathbf{L})^{-1}$ is known as the Tikhonov graph filter with frequency response $h_t(\lambda) = 1/(1+\mu\lambda)$. Applications of the Tikhonov filter in graph signal processing include signal denoising [2], classification [7], and inter-predicted video coding [31].
- *Bandpass exponential filter:* bandpass graph filters are key components in M-channel graph filter banks [58], [59]. Here, we consider the frequency response

$$h_{\exp}(\lambda) = \exp(-\gamma(\lambda - \lambda_{pb})^2),$$

  where $\gamma > 0$ and λ_{pb} is the central frequency of the passband.

For the choice of parameters, we use $\mu = 0.25$, $\gamma = 1$, and $\lambda_c = \lambda_{pb} = 0.5\lambda_{max}$ in this experiment. The following filter implementations are compared:

- *Polynomial DCT filter:* given the desired frequency response, two implementation methods for PGF (with LS design) are considered, namely, *PGF-i*, the iterative implementation described in Sec. II-B and *PGF-C* which implements PGFs using recurrence relations of Chebyshev polynomials [18].
- *Multivariate polynomial DCT filter:* we consider all sparse graph operators (289 operators for the 16×16 grid and 65 operators for the length-64 line graph). Then, we obtain the least squares filter (18) with an ℓ_0 constraint and $K = 1$ using orthogonal matching pursuit, with R being 2 to 8.
- *Autoregressive moving average (ARMA) graph filter [12]*: we consider an IIR graph filter in rational polynomial form, i.e.,

$$\mathbf{H}_{\text{ARMA}} = \left(\sum_{p=0}^{P} a_p \mathbf{Z}^p \right)^{-1} \left(\sum_{q=0}^{Q} b_q \mathbf{Z}^q \right).$$

[3]The source code for this experiment is available in [56].

We choose polynomial degrees as $Q = P = 2$ and consider different numbers of iterations T. The graph filter implementation is based on the conjugate gradient approach described in [21], whose complexity is $\mathcal{O}((PT+Q)E)$.

- *Exact filter with fast DCT:* the filter operation is performed by a cascade of a forward DCT, a frequency masking with h, and an inverse DCT, where the forward and inverse DCTs are implemented using well-known fast algorithms [60]. For 4×4 or 16×16 grids, 2D separable DCTs are implemented, where a fast 1D DCT is applied to all rows and columns.

In LS designs, uniform weights $\boldsymbol{\rho} = \mathbf{1}$ are used. For each graph we consider, 20000 random input signals are generated and the complexity for each graph filter method is evaluated as an average runtime over all 20000 trials. We measure the error between approximate and exact frequency responses with the root normalized mean square error $\|\mathbf{h}_{\text{approx}} - \mathbf{h}\|/\|\mathbf{h}\|$.

We show in Fig. 8 the resulting runtimes and errors, where a point closer to the origin correspond to a better trade-off between complexity and approximation accuracy. We observe in Fig. 8(a)(c) that low degree PGFs accurately approximate the Tikhonov filter, whose frequency response is closer to a linear function of λ. In Fig. 8(b)(d), for bandpass exponential filter on the length-64 line graph, MPGF achieves a higher accuracy with lower complexity than PGF and ARMA graph filters. As discussed in Sec. VI-C, the complexity of PGF and MPGF grows linearly with the graph size, while the fast DCT algorithm has $\mathcal{O}(N \log N)$ complexity. Thus, PGF and MPGF would achieve a better speed performance with respect to exact filter when the graph size is larger. Note that in this experiment, a fast algorithm with $\mathcal{O}(N \log N)$ complexity for the GFT (DCT-II) is available. However, this is not always true for arbitrary graph size N, nor for other types of DTTs, where fast exact graph filter may not be available.

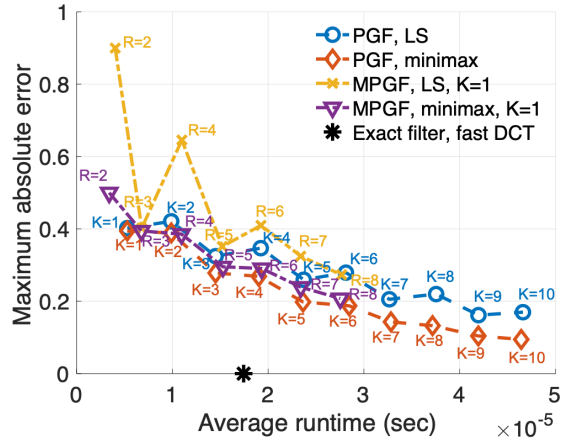
2) Evaluation of minimax designs.: Next, we consider an ideal low-pass filter:

$$h_{LP}(\lambda) = \begin{cases} 1, & 0 \le \lambda \le \lambda_c \\ 0, & \text{otherwise} \end{cases}$$

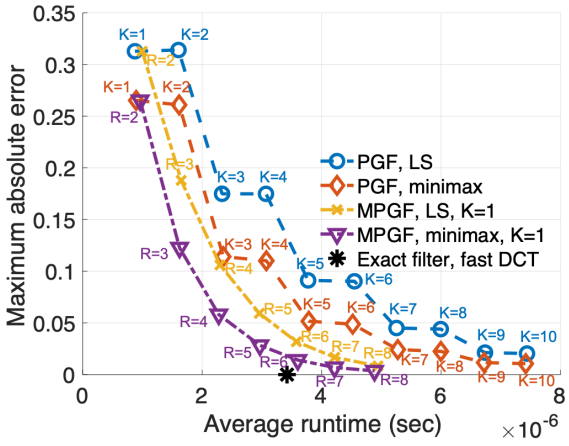
where $\lambda_c = 0.5\lambda_{max}$ is the cut-off frequency. The weight ρ_i is chosen to be 0 in the transition band $0.4 \le \lambda_i \le 0.6$, and 1 in passband and stopband. Fig. 9 shows the resulting runtimes and approximation errors, which are measured with the maximum absolute error between approximate and desired frequency responses in passband and stopband: $\max_i \rho_i |h_{\text{approx}}(\lambda_i) - h(\lambda_i)|$. We can see in Fig. 9 that, when K or R increases, the maximum absolute error steadily decreases in PGF and MPGF designs with minimax criteria. In contrast, PGF and MPGF designs with LS criterion may lead to non-monotonic behavior in terms of the maximum absolute error as in Fig. 9(a). In fact, under the LS criterion, using more sparse operators will reduce the least squares error, but does not always decrease the maximum absolute error.

Based on the results in Figs. 8 and 9, we provide some remarks on the choice of DTT filter implementation:

- If the desired frequency response is close to a linear function of λ, e.g., Tikhonov filters with a small μ or graph diffusion processes [61], then a low-order PGF would be sufficiently accurate, and has the lowest complexity.
- If the graph size is small, transform length allows a fast DTT algorithm, or when separable DTTs are available (e.g., on a 16×16 grid), DTT filter with fast DTT implementation would be favorable.
- For a sufficiently large length (e.g., $N = 64$) and a frequency response that is non-smooth (e.g., ideal low-pass filter) or non-monotonic (e.g., bandpass filter), an MPGF design may fit the desired filter with a reasonable speed performance. In particular, we note that $\mathbf{Z}^{(2)}$ is a bandpass filter with passband center $\lambda_{pb} = \lambda_{max}/2$. Thus, MPGF using $\mathbf{Z}^{(2)}$ would provide an efficiency improvement for bandpass filters with λ_{pb} close to $\lambda_{max}/2$.
- When robustness of the frequency response in the maximum absolute error sense is an important concern, a design based on minimax criterion would be preferable.

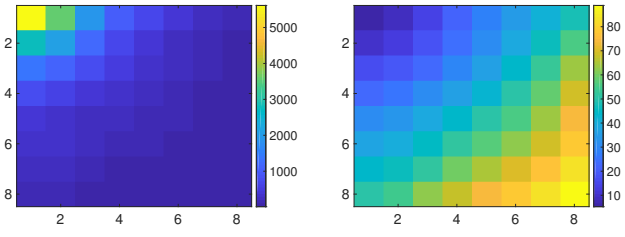


(a) Ideal low-pass, 16×16 grid



(b) Ideal low-pass, length-64 line graph

Fig. 9. Runtime vs maximum absolute error for various designs of ideal low-pass filter on (a) 16×16 grid, and (b) length-64 line graph.



(a) (b)

Fig. 10. Statistics of 8×8 AV1 transform coefficients collected from `BasketballDrill` and `BQMall` clips. (a) Average energy of each transform coefficients. (b) Percentage of coefficients whose absolute values are smaller than an energy threshold $\tau = 5$.

B. Transform Type Selection in Video Coding

In the second experiment, we consider the quadratic form (25) as a transform type cost, and apply the method described in Sec. VI-B to speed up transform type selection in the AV1 codec [43]. In transform coding [62], (25) can be used as a proxy of the bitrate cost for block-wise transform type optimization [36], [37]. In particular, we denote $\mathbf{x}$ an image or video block, and $\mathbf{\Phi}$ the orthogonal transform applied to $\mathbf{x}$. Lower bitrate cost can be achieved if $\mathbf{\Phi}$ gives a high energy compaction in the low frequencies, i.e., the energy of $\mathbf{\Phi}^\top\mathbf{x}$ is concentrated in the first few entries. Thus, the proxy of cost (25) can be defined with positive and increasing $\mathbf{q}$ ($0 < q_1 < \cdots < q_N$) to penalize large and high frequency coefficients, thus favoring transforms having more energy in the low frequencies. To verify this, we demonstrate some statistics of AV1 sample blocks in Figure 10, where we observe that the average energy of transform coefficients monotonically decreases from lower to higher frequencies. In addition, entropy coding in AV1 also includes features for encoding lower frequency coefficients with fewer bits [63]. Based on the above observations, we will choose the weights as increasing functions.

AV1 includes four 1D transforms: 1) $\mathbf{U}$: DCT, 2) $\mathbf{V}$: ADST, 3) $\mathbf{JV}$: FLIPADST, which has flipped ADST functions, and 4) $\mathbf{I}$: IDTX (identity transform), where no transform will be applied. For small inter predicted blocks, all 2D combinations of 1D transforms are used. Namely, there are 16 2D transforms candidates, $(\mathbf{T}_{\text{col}}, \mathbf{T}_{\text{row}})$ with $\mathbf{T}_{\text{col}}, \mathbf{T}_{\text{row}} \in \{\mathbf{U}, \mathbf{V}, \mathbf{JV}, \mathbf{I}\}$, which makes the encoder computationally expensive. Recent work on encoder complexity reduction includes [23], [64], [65], which apply heuristic and data-driven techniques to prune transform types during the search.

To speed up transform type selection in AV1, for 1D pixel block $\mathbf{x} \in \mathbb{R}^N$, we choose the following increasing weights for (25)[4]:

$$q_i = \delta_i = 2 - 2\cos\left(\frac{(i-\frac{1}{2})\pi}{N}\right). \tag{29}$$

Then, different transform type costs would be given by (25) with different $\mathbf{\Phi}$, i.e., $\mathcal{C}_{\mathbf{T}}(\mathbf{x};\mathbf{q})$ with $\mathbf{T} \in \{\mathbf{U}, \mathbf{V}, \mathbf{JV}, \mathbf{I}\}$. This choice allows efficient computation of exact $\mathcal{C}_{\mathbf{V}}(\mathbf{x};\mathbf{q})$ and $\mathcal{C}_{\mathbf{JV}}(\mathbf{x};\mathbf{q})$ through their corresponding sparse Laplacian matrices:

$$\mathcal{C}_{\mathbf{V}}(\mathbf{x};\mathbf{q}) = \mathbf{x}^\top\mathbf{L}_{\text{A}}\mathbf{x}, \quad \mathcal{C}_{\mathbf{JV}}(\mathbf{x};\mathbf{q}) = \mathbf{x}^\top\mathbf{J}\mathbf{L}_{\text{A}}\mathbf{J}\mathbf{x},$$

where $\mathbf{J}\mathbf{L}_{\text{A}}\mathbf{J}$ is the left-right and up-down flipped version of $\mathbf{L}_{\text{A}}$. For the approximation of DCT cost $\mathcal{C}_{\mathbf{U}}(\mathbf{x};\mathbf{q})$, we obtain $R = 3$ nonzeros polynomial coefficients g_m with degree $L = 1$ as in (28) using an exhaustive search. As a result, costs for all 1D transforms can be computed in the pixel domain as follows

$$\begin{aligned}
\mathcal{Q}_{\mathbf{U}} &= \mathbf{x}_i^\top\left(g_0\mathbf{I} + \sum_{m=1}^{M} g_m\mathbf{Z}_{\text{DCT-II}}^{(m)}\right)\mathbf{x}_i \\
\mathcal{Q}_{\mathbf{V}} &= \mathcal{C}_{\mathbf{V}}(\mathbf{x}_i;\mathbf{q}) = \mathbf{x}_i^\top\mathbf{L}_{\text{A}}\mathbf{x}_i \\
\mathcal{Q}_{\mathbf{JV}} &= \mathcal{C}_{\mathbf{JV}}(\mathbf{x}_i;\mathbf{q}) = \mathbf{x}_i^\top\mathbf{J}\mathbf{L}_{\text{A}}\mathbf{J}\mathbf{x}_i \\
\mathcal{Q}_{\mathbf{I}} &= \mathcal{C}_{\mathbf{I}}(\mathbf{x}_i;\mathbf{q}) = \sum_j w_j\mathbf{x}_i(j)^2,
\end{aligned} \tag{30}$$

where M is the number of DCT operators and g_m has only $R = 3$ non-zero elements.

[4]As (25) is used a proxy of the actual bitrate cost, we leave out the search of optimal weights. The weights in (29) are used because of the computational convenience for $\mathcal{Q}_V = \mathbf{x}_i^\top\mathbf{L}_{\text{A}}\mathbf{x}_i$. In fact, additional experiments with a set of weights obtained from least squares regression with respect to the actual bitrates extracted from AV1, led to only marginal improvements in quality. The weights from (29) still lead to a better speed-quality trade-off. A possible reason for this outcome is that the weighted square sum is, in any case, not a very accurate proxy for the bitrate. Thus, further improving the quality loss would be a challenging task, even with weights that are learned from the data.

TABLE IV
ENCODING TIME AND QUALITY LOSS (IN BD RATE) OF DIFFERENT TRANSFORM PRUNING METHODS. THE BASELINE IS AV1 WITH A FULL TRANSFORM SEARCH (NO PRUNING). A SMALLER LOSS IS BETTER.

Method	Encoding time	Quality loss
PRUNE_LAPLACIAN [23]	91.71%	0.32%
PRUNE_OPERATOR	89.05%	0.31%
PRUNE_2D_FAST [64]	86.78%	0.05%

TABLE V
ENCODING TIME AND QUALITY LOSS (IN BD RATE) OF PRUNE_OPERATORS VERSUS PRUNE_2D_FAST. SMALLER OR NEGATIVE LOSS IS BETTER.

Sequence	Encoding time	Quality loss
akiyo	102.10%	0.00%
bowing	97.22%	-0.14%
bus	103.92%	-0.17%
city	102.36%	0.18%
crew	103.65%	0.07%
foreman	104.29%	0.07%
harbour	106.49%	-0.06%
ice	105.22%	0.30%
mobile	103.27%	0.23%
news	103.29%	-0.09%
pamphlet	97.75%	0.21%
paris	105.54%	0.21%
soccer	104.53%	0.22%
students	100.71%	0.03%
waterfall	102.34%	0.23%
Overall	102.61%	0.26%

Extending our previous experiment PRUNE_LAPLACIAN in [23], we implemented a new experiment named PRUNE_OPERATORS in AV1[5]. We implement the integer versions of the transform cost evaluation (30) for transform lengths 4, 8, 16, and 32. Within each 2D block, we take an average over all columns or rows, to obtain column and row costs $\mathcal{Q}_{\mathbf{T}}^{(\text{col})}$ and $\mathcal{Q}_{\mathbf{T}}^{(\text{row})}$ with $\mathbf{T} \in \{\mathbf{U}, \mathbf{V}, \mathbf{JV}, \mathbf{I}\}$. Those costs are aggregated into 16 2D transform costs by summing the associated column and row costs. For example, the cost associated to vertical ADST and horizontal DCT is given by

$$\mathcal{Q}_{(\mathbf{V},\mathbf{U})} = \mathcal{Q}_{\mathbf{V}}^{(\text{col})} + \mathcal{Q}_{\mathbf{U}}^{(\text{row})}.$$

Finally, we design a pruning criteria, where each 2D column (or row) transform will be pruned if its associated cost is relatively large compared to the others.

C1. For $\mathbf{T}_{\text{col}}, \mathbf{T}_{\text{row}} \in \{\mathbf{U}, \mathbf{V}, \mathbf{JV}\}$, prune $(\mathbf{T}_{\text{col}}, \mathbf{T}_{\text{row}})$ if

$$\begin{aligned}
&\mathcal{Q}_{(\mathbf{T}_{\text{col}},\mathbf{T}_{\text{row}})} > \\
&\quad \tau_1\left(\mathcal{Q}_{\mathbf{U}}^{(\text{col})} + \mathcal{Q}_{\mathbf{V}}^{(\text{col})} + \mathcal{Q}_{\mathbf{JV}}^{(\text{col})} + \mathcal{Q}_{\mathbf{U}}^{(\text{row})} + \mathcal{Q}_{\mathbf{V}}^{(\text{row})} + \mathcal{Q}_{\mathbf{JV}}^{(\text{row})}\right).
\end{aligned}$$

C2. For $\mathbf{T}_{\text{col}} = \mathbf{I}$ or $\mathbf{T}_{\text{row}} = \mathbf{I}$, prune $(\mathbf{T}_{\text{col}}, \mathbf{T}_{\text{row}})$ if

$$\begin{aligned}
\mathcal{Q}_{(\mathbf{T}_{\text{col}},\mathbf{T}_{\text{row}})} > \quad &\tau_2\left(\mathcal{Q}_{\mathbf{U}}^{(\text{col})} + \mathcal{Q}_{\mathbf{V}}^{(\text{col})} + \mathcal{Q}_{\mathbf{JV}}^{(\text{col})} + \mathcal{Q}_{\mathbf{I}}^{(\text{col})}\right. \\
&\left. + \mathcal{Q}_{\mathbf{U}}^{(\text{row})} + \mathcal{Q}_{\mathbf{V}}^{(\text{row})} + \mathcal{Q}_{\mathbf{JV}}^{(\text{row})} + \mathcal{Q}_{\mathbf{I}}^{(\text{row})}\right).
\end{aligned}$$

where threshold parameters are chosen as $\tau_1 = 0.34$,$\tau_2 = 0.33$. Note that the number of 1D transforms being pruned can be different for different blocks. The pruning rules C1 do not depend on $\mathcal{Q}_{\mathbf{I}}$ because IDTX tends to have a larger bitrate cost with a significantly lower computational complexity than the other transforms. Thus, more aggressive pruning criteria C1 is applied to $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{JV}$ to reduce more encoding time.

This pruning scheme is evaluated using 15 benchmark test sequences: akiyo, bowing, bus, city, crew, foreman,

[5]The experiment has been implemented on a version in July 2020. Available: https://aomedia-review.googlesource.com/c/aom/+/113461

harbour, ice, mobile, news, pamphlet, paris, soccer, students, and waterfall. The results are shown in Table IV, where the speed improvement is measured in the percentage of encoding time compared to the scheme without any pruning. Each number in the table is an average over several target bitrate levels: 300, 600, 1000, 1500, 2000, 2500, and 3000 kbps. Note that the proposed method yields a smaller quality loss with shorter encoding time than in our previous work [23]. Our method does not outperform the state-of-the-art methods PRUNE_2D_FAST in terms of the average BD rate, but shows a gain in particular video sequences such as bowing (as shown in Table V. Note that in [64], for each supported block size ($N \times N$, $N \times 2N$ and $2N \times N$, with $N \in \{4, 8, 16\}$), a specific neural network is required to obtain the scores, involving more than 5000 parameters to be learned in total. In contrast, our approach only requires the weights $\mathbf{q}$ to be determined for each transform length, requiring $4 + 8 + 16 + 32 = 60$ parameters. With or without optimized weights, our model is more interpretable than the neural-network-based model, as has a significantly smaller number of parameters, whose meaning can be readily explained.

VIII. Conclusion

In this work we explored discrete trigonometric transform (DTT) filtering approaches using sparse graph operators. First, we introduced fundamental graph operators associated to 8 DCTs and 8 DSTs by exploiting trigonometric properties of their transform bases. We also showed that these sparse operators can be extended to 2D separable transforms involving 1D DTTs. Considering a weighted setting for frequency response approximation, we proposed least squares and minimax approaches for both polynomial graph filter (PGF) and multivariate polynomial graph filter (MPGF) designs. We demonstrated through an experiment that PGF and MPGF designs would provide a speedup compared to traditional DTT filter implemented in transform domain. We also used MPGF to design a speedup technique for transform type selection in a video encoder, where a significant complexity reduction can be obtained.

Appendix A

This appendix presents brief derivations for sparse operators of DST-IV, DST-VII and DCT-V.

A. Sparse DST-IV Operators

Recall the definition of DST-IV functions as in (8):

$$\phi_j(k) = v_j(k) = \sqrt{\frac{2}{N}} \sin \frac{(j-\frac{1}{2})(k-\frac{1}{2})\pi}{N}$$

As in Sec. III-A, we can obtain

$$\begin{aligned} &v_j(p-\ell) + v_j(p+\ell) \\ &= \sqrt{\frac{2}{N}} \left[\sin \frac{(j-\frac{1}{2})(p-\ell-\frac{1}{2})\pi}{N} + \sin \frac{(j-\frac{1}{2})(p-\ell-\frac{1}{2})\pi}{N} \right] \\ &= 2\sqrt{\frac{2}{N}} \sin \frac{(j-\frac{1}{2})(p-\ell-\frac{1}{2})\pi}{N} \cos \frac{\ell(j-\frac{1}{2})\pi}{N} \\ &= \left(2\cos \frac{\ell(j-\frac{1}{2})\pi}{N} \right) v_j(p), \end{aligned}$$

where we have applied the trigonometric identity

$$\sin\alpha + \sin\beta = 2\sin\left(\frac{\alpha+\beta}{2}\right)\cos\left(\frac{\alpha-\beta}{2}\right). \tag{31}$$

By the left and right boundary condition of DST-IV, we have

$$v_j(p-\ell) = -v_j(-p+\ell+1), \quad v_j(p+\ell) = v_j(-p-\ell+2N+1).$$

which gives the following result:

Proposition 4. *For $\ell = 1, \ldots, N-1$, we define $\mathbf{Z}^{(\ell)}_{DST\text{-}IV}$ as a $N \times N$ matrix, whose p-th row has only two non-zero elements specified as follows:*

$$\left(\mathbf{Z}^{(\ell)}_{DST\text{-}IV}\right)_{p,q_1} = \begin{cases} 1 \text{ with } q_1 = p-\ell, & \text{if } p-\ell \geq 1 \\ -1 \text{ with } q_1 = -p+\ell+1, & \text{otherwise} \end{cases},$$

$$\left(\mathbf{Z}^{(\ell)}_{DST\text{-}IV}\right)_{p,q_2} = 1, \quad q_2 = \begin{cases} p+\ell, & \text{if } p+\ell \leq N \\ -p-\ell+2N+1, & \text{otherwise} \end{cases}$$

The corresponding eigenvalues are $\lambda_j = 2\cos\frac{\ell(j-\frac{1}{2})\pi}{N}$.

B. Sparse DST-VII Operators

Now, we consider the basis function of DST-VII:

$$\phi_j(k) = \frac{2}{\sqrt{2N+1}} \sin \frac{\left(j-\frac{1}{2}\right)k\pi}{N+\frac{1}{2}}.$$

Then, by (31) we have

$$\begin{aligned} &\phi_j(p-\ell) + \phi_j(p+\ell) \\ &= \frac{2}{\sqrt{2N+1}} \left[\sin \frac{\left(j-\frac{1}{2}\right)(p-\ell)\pi}{N+\frac{1}{2}} + \sin \frac{\left(j-\frac{1}{2}\right)(p+\ell)\pi}{N+\frac{1}{2}} \right] \\ &= \frac{2}{\sqrt{2N+1}} 2\sin \frac{\left(j-\frac{1}{2}\right)p\pi}{N+\frac{1}{2}} \cos \frac{\ell\left(j-\frac{1}{2}\right)\pi}{N+\frac{1}{2}} \\ &= \left(2\cos \frac{\ell\left(j-\frac{1}{2}\right)\pi}{N+\frac{1}{2}} \right) \phi_j(p) \end{aligned} \tag{32}$$

The left boundary condition (i.e., $\phi_j(k) = -\phi_j(-k)$) of DST-VII corresponds to $\phi_j(p-\ell) = -\phi_j(-p+\ell)$. Together with the right boundary condition $\phi_j(p+\ell) = \phi_j(-p-\ell+2N+1)$, we have the following proposition.

Proposition 5. *For $\ell = 1, \ldots, N-1$, we define $\mathbf{Z}^{(\ell)}_{DST\text{-}VII}$ as a $N \times N$ matrix, whose p-th row has at most two non-zero elements specified as follows:*

$$\left(\mathbf{Z}^{(\ell)}_{DST\text{-}VII}\right)_{p,q_1} = \begin{cases} 1 \text{ with } q_1 = p-\ell, & \text{if } p > \ell \\ -1 \text{ with } q_1 = -p+\ell, & \text{if } p < \ell \end{cases},$$

$$\left(\mathbf{Z}^{(\ell)}_{DST\text{-}VII}\right)_{p,q_2} = 1, \quad q_2 = \begin{cases} p+\ell, & \text{if } p+\ell \leq N \\ -p-\ell+2N+1, & \text{otherwise} \end{cases}$$

The corresponding eigenvalues are $\lambda_j = 2\cos\frac{\ell(j-\frac{1}{2})\pi}{N+\frac{1}{2}}$.

In Proposition 5, note that the ℓ-th row has only one nonzero element because when $p = \ell$, $\phi_j(p-\ell) = 0$, and (32) reduces to

$$\phi_j(p+\ell) = \left(2\cos \frac{\ell\left(j-\frac{1}{2}\right)\pi}{N+\frac{1}{2}} \right) \phi_j(p).$$

C. Sparse DCT-V Operators

Here, ϕ_j are defined as DCT-V basis functions

$$\phi_j(k) = \frac{2}{\sqrt{2N-1}} c_j c_k \sin \frac{(j-1)(k-1)\pi}{N-\frac{1}{2}}.$$

Note that $c_k = 1/\sqrt{2}$ for $k = 1$ and 1 otherwise. For the trigonometric identity (15) to be applied, we introduce a scaling factor such that $b_k c_k = 1$ for all k:

$$b_k = \begin{cases} \sqrt{2}, & k = 1 \\ 1, & \text{otherwise} \end{cases}.$$

In this way, by (15) we have

$$\begin{aligned}
&b_{p-\ell}\cdot\phi_j(p-\ell)+b_{p+\ell}\cdot\phi_j(p+\ell)\\
&=\frac{2}{\sqrt{2N-1}}c_j\left[\cos\frac{(j-1)(p-\ell-1)\pi}{N-\frac{1}{2}}+\cos\frac{(j-1)(p+\ell-1)\pi}{N-\frac{1}{2}}\right]\\
&=\frac{2}{\sqrt{2N-1}}c_j 2\cos\frac{(j-1)(p-1)\pi}{N-\frac{1}{2}}\cos\frac{\ell(j-1)\pi}{N-\frac{1}{2}}\\
&=\left(2\cos\frac{\ell(j-1)\pi}{N-\frac{1}{2}}\right)b_p\phi_j(p),
\end{aligned}$$

so this eigenvalue equation can be written as

$$\frac{b_{p-\ell}}{b_p}\cdot\phi_j(p-\ell)+\frac{b_{p+\ell}}{b_p}\cdot\phi_j(p+\ell)=\left(2\cos\frac{\ell(j-1)\pi}{N-\frac{1}{2}}\right)\phi_j(p). \quad (33)$$

The left boundary condition of DCT-V corresponds to $\phi_j(p-\ell)=\phi_j(-p+\ell+2)$, and the right boundary condition gives $\phi_j(p+\ell)=\phi_j(-p-\ell+2N+1)$. Thus, (33) yields the following proposition:

Proposition 6. *For $\ell=1,\dots,N-1$, we define $\mathbf{Z}^{(\ell)}_{DCT\text{-}V}$ as a $N\times N$ matrix, whose p-th row has at most two non-zero elements specified as follows:*

$$\left(\mathbf{Z}^{(\ell)}_{DCT\text{-}V}\right)_{p,q_1}=\begin{cases}\sqrt{2}\ \textit{with}\ q_1=1, & \textit{if}\ p-\ell=1\\ 1\ \textit{with}\ q_1=p-\ell, & \textit{if}\ p-\ell>1\\ 1\ \textit{with}\ q_1=-p+\ell+2, & \textit{if}\ p-\ell\le 0, p\ne 1\end{cases},$$

$$\left(\mathbf{Z}^{(\ell)}_{DCT\text{-}V}\right)_{p,q_2}=\begin{cases}\sqrt{2}\ \textit{with}\ q_2=1, & p=1\\ 1\ \textit{with}\ q_2=p+\ell, & p\ne 1, p+\ell\le N\\ 1\ \textit{with}\ q_2=-p-\ell+2N+1, & \textit{otherwise}\end{cases}$$

The corresponding eigenvalues are $\lambda_j=2\cos\frac{\ell(j-1)\pi}{N-\frac{1}{2}}$.

The values of $\sqrt{2}$ in Proposition 6 arise from $b_{p-\ell}/b_p$ and $b_{p+\ell}/b_p$ in the LHS of (33). In particular, when $p=\ell+1$ we have $b_{p-\ell}/b_p=\sqrt{2}$ and $b_{p+\ell}/b_p=1$, so (33) gives

$$\sqrt{2}\phi_j(p-\ell)+\phi_j(p+\ell)=\left(2\cos\frac{\ell(j-1)\pi}{N-\frac{1}{2}}\right)\phi_j(p).$$

When $p=1$, $b_{p-\ell}/b_p=b_{p+\ell}/b_p=1/\sqrt{2}$. In addition, by the left boundary condition, $\phi_j(p-\ell)=\phi_j(p+\ell)$, so (33) reduces to

$$\sqrt{2}\phi_j(p+\ell)=\left(2\cos\frac{\ell(j-1)\pi}{N-\frac{1}{2}}\right)\phi_j(p),\quad \text{for } p=1.$$

meaning that the first row of $\mathbf{Z}^{(\ell)}_{\text{DCT-V}}$ has one nonzero element only.

Appendix B

This appendix includes remarks on the retrieval of sparse operators for general GFTs beyond DCT and DST.

The characterization of all Laplacians that share a common GFT has been studied in the context of graph topology identification and graph diffusion process inference [52], [66], [67]. In particular, it has been shown in [52] that the set of *normalized* Laplacian matrices having a fixed GFT can be characterized by a convex polytope. Following a similar proof, we briefly present the counterpart result for *unnormalized* Laplacian with self-loops allowed:

Theorem 1. *The set of Laplacian matrices with a fixed GFT can be characterized by a convex polyhedral cone in the space of eigenvalues $(\lambda_1,\dots,\lambda_N)$.*

Proof: For a given GFT $\mathbf{\Phi}$, let the eigenvalues λ_j of the Laplacian be variables. By definition of the Laplacian (4), we can see that

$$\mathbf{L}=\mathbf{\Phi}\cdot\mathrm{diag}(\lambda_1,\dots,\lambda_N)\cdot\mathbf{\Phi}^\top=\sum_{k=1}^N\lambda_k\boldsymbol{\phi}_k\boldsymbol{\phi}_k^\top \quad (34)$$

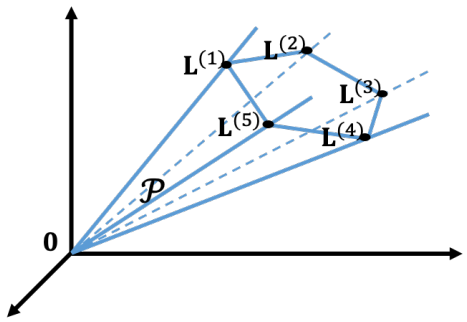


Fig. 11. An illustrative example of a polyhedral cone in $\mathbb{R}^3$ with a vertex at $\mathbf{0}$ and 5 edges. Any element of the cone can be represented as $\sum_{m=1}^5 a_m\mathbf{L}^{(m)}$ with non-negative a_m.

is a valid Laplacian matrix if $l_{ij}\le 0$ (non-negative edge weights), $l_{ii}\ge\sum_{j=1,j\ne i}^N l_{ij}$ (non-negative self-loop weights), and $\lambda_k\ge 0$ for all k (non-negative graph frequencies). With the expression (34) we have $l_{ij}=\sum_{k=1}^N\lambda_k\phi_k(i)\phi_k(j)$, and thus the Laplacian conditions can be expressed in terms of λ_j's:

$$\begin{aligned}
&\sum_{k=1}^N\lambda_k\phi_k(i)\phi_k(j)\le 0,\quad \text{for } i\ne j,\\
&\sum_{k=1}^N\lambda_k\phi_k(i)^2\ge\sum_{\substack{j=1\\ j\ne i}}^N\lambda_k\phi_k(i)\phi_k(j),\quad \text{for } i=1,\dots,N,\\
&\lambda_k\ge 0,\quad k=1,\dots,N.
\end{aligned} \quad (35)$$

These constraints on $\boldsymbol{\lambda}=(\lambda_1,\dots,\lambda_N)^\top$ are all linear, so the feasible set for $\boldsymbol{\lambda}\in\mathbb{R}^N$ is a convex polyhedron. We denote this polyhedron by $\mathcal{P}$, and highlight some properties as follows:

- $\mathcal{P}$ is non-empty: it is clear to see that $\boldsymbol{\lambda}=\mathbf{1}$ gives $\mathbf{L}=\mathbf{I}$, which is a trivial, but valid, Laplacian.
- $\boldsymbol{\lambda}=\mathbf{0}$ is the only vertex of $\mathcal{P}$: when $\lambda_j=0$ for all j, equality is met for all constraints in (35). This means that all hyperplanes that define $\mathcal{P}$ intersect at a common point $\mathbf{0}$, which further implies that $\mathcal{P}$ does not have other vertices than $\mathbf{0}$.

From those facts above, we conclude that $\mathcal{P}$ is a non-empty convex polyhedral cone. □

For illustration purpose, we can visualize the structure of a 3-dimensional polyhedral cone with 5 edges in Fig. 11. Notably, any element in $\mathcal{P}$ can be expressed by a conical combination (linear combination with non-negative coefficients) of elements on the edges of $\mathcal{P}$, as illustrated in Fig. 11. In particular, let $\mathcal{P}$ have M edges, and let $\mathbf{L}^{(1)},\dots,\mathbf{L}^{(M)}$ be points on different edges, then any element $\mathbf{Q}\in\mathcal{P}$ can be represented as

$$\mathbf{Q}=\sum_{m=1}^M a_m\mathbf{L}^{(m)},\quad a_m\ge 0.$$

The fact that Laplacians have non-positive off-diagonal entries implies that the $\mathbf{L}^{(m)}$'s are the most sparse Laplacians. This can be seen by noting that a conical combination of two Laplacians must have more non-zero off-diagonal elements than the two individual Laplacians do.

Since sparse Laplacians are characterized by edges of a polyhedral cone, we can choose sparse operators in (3) as those Laplacians: $\mathcal{Z}=\{\mathbf{L}^{(k)}\}_k$. The retrieval of those matrices would require an algorithm that enumerates the vertices and edges given the description of a polyhedron. A popular algorithm for this problem is the so-called reverse search [68], which has a complexity $\mathcal{O}(rdv)$, where r is the number of linear constraints in $\mathbb{R}^d$, and v is the number of target

vertices. In (35), $d = N$ and $m = (N^2 + 3N)/2$, so the complexity reduces to $\mathcal{O}(N^3 v)$. In practice, the vertex enumeration problem is in general an NP-hard problem since the number of vertices v can be a combinatorial number: $\binom{r}{d}$. For the purpose of efficient graph filter design, a truncated version of the algorithm [68] may be applied to obtain a few instead of all vertices. The study of such a truncated algorithm will be left for our future work.

APPENDIX C

This appendix shows a construction of sparse operator for graphs with certain symmetry properties. In our recent work [11], we highlighted that a GFT has a butterfly stage for fast implementation if the associated graph demonstrates a symmetry property based on involution permutation (pairing function of nodes):

Definition 1. *A permutation φ on a finite set $\mathcal{V}$ is an involution if $\varphi(\varphi(i)) = i,\ \forall i \in \mathcal{V}$.*

Definition 2 ([11])**.** *Given an involution φ on the vertex set $\mathcal{V}$ of graph $\mathcal{G}$, then $\mathcal{G}$, with a weighted adjacency matrix $\mathbf{W}$, is call φ-symmetric if $w_{i,j} = w_{\varphi(i),\varphi(j)},\ \forall i \in \mathcal{V}, j \in \mathcal{V}$.*

With a φ-symmetric graph $\mathcal{G}$, a sparse operator can be constructed as follows.

Lemma 2. *Given a φ-symmetric graph $\mathcal{G}$ with Laplacian $\mathbf{L}$, we can construct a graph $\overline{\mathcal{G}_\varphi}$ by connecting nodes i and j with edge weight 1 for all node pairs (i, j) with $\varphi(i) = j, i \neq j$. In this way, the Laplacian $\overline{\mathbf{L}_\varphi}$ of $\overline{\mathcal{G}_\varphi}$ commutes with $\mathbf{L}$.*

Proof: We note that, for $i \in \mathcal{V}$, we either have $\varphi(i) = j \neq i$ with $\varphi(j) = i$ or $\varphi(i) = i$. Without loss of generality, we order the graph vertices such that $\varphi(i) = N + 1 - i$ for $i = 1, \ldots, k$ and $\varphi(i) = i$ for $i = k+1, \ldots, N+1-k$. With this vertex order, we express $\mathbf{L}$ in terms of block matrix components,

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_{11} & \mathbf{L}_{12} & \mathbf{L}_{13} \\ \mathbf{L}_{12}^\top & \mathbf{L}_{22} & \mathbf{L}_{23} \\ \mathbf{L}_{13}^\top & \mathbf{L}_{23}^\top & \mathbf{L}_{33} \end{pmatrix},$$

where $\mathbf{L}_{11}, \mathbf{L}_{33} \in \mathbb{R}^{k\times k}$ and $\mathbf{L}_{22} \in \mathbb{R}^{(N-2k)\times(N-2k)}$. By φ-symmetry, the block components of $\mathbf{L}$ satisfy ([11, Lemma 3])

$$\mathbf{L}_{13} = \mathbf{J}\mathbf{L}_{13}^\top\mathbf{J}, \quad \mathbf{L}_{33} = \mathbf{J}\mathbf{L}_{11}\mathbf{J}, \quad \mathbf{L}_{23} = \mathbf{L}_{12}^\top\mathbf{J}. \tag{36}$$

We can also see that the Laplacian constructed from Lemma 2, with the same node ordering defined as above, is

$$\overline{\mathbf{L}_\varphi} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & -\mathbf{J} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\mathbf{J} & \mathbf{0} & \mathbf{I} \end{pmatrix}.$$

Then, using (36), we can easily verify that

$$\mathbf{L}\overline{\mathbf{L}_\varphi} = \begin{pmatrix} \mathbf{L}_{11} - \mathbf{L}_{13}\mathbf{J} & \mathbf{0} & -\mathbf{L}_{11}\mathbf{J} + \mathbf{L}_{13} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{L}_{13}^\top - \mathbf{J}\mathbf{L}_{11} & \mathbf{0} & -\mathbf{L}_{13}^\top\mathbf{J} + \mathbf{J}\mathbf{L}_{11}\mathbf{J} \end{pmatrix} = \overline{\mathbf{L}_\varphi}\mathbf{L},$$

which concludes the proof. □

We demonstrate an example for the construction of $\overline{\mathcal{G}_\varphi}$, in Fig. 12. Fig. 12(a) shows a 15-node human skeletal graph $\mathcal{G}$ [69]. A left-to-right symmetry can be observed in $\mathcal{G}$, which induces an involution φ with $\varphi(i) = i$ for $i = 7, 8, 9$ and $\varphi(i) = 16 - i$ otherwise. With the construction in Lemma 2, we obtain a graph $\overline{\mathcal{G}_\varphi}$ as in Fig. 12(b) by connecting all pairs of symmetric nodes in Fig. 12(a). We denote $\mathbf{Z}^{(1)} = \mathbf{L}$ and $\mathbf{Z}^{(2)} = \overline{\mathbf{L}_\varphi}$ the Laplacians of $\mathcal{G}$ and $\overline{\mathcal{G}_\varphi}$, respectively,

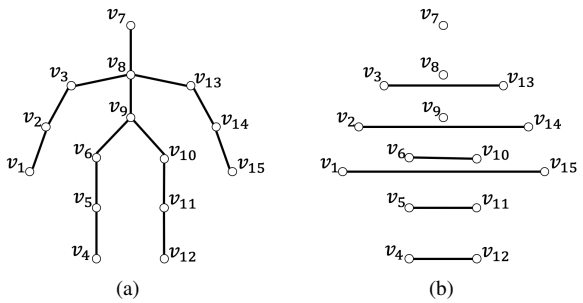


Fig. 12. An illustrative example for graph operator construction based on graph symmetry. (a) The 15-node human skeletal graph $\mathcal{G}$. (b) The graph $\overline{\mathcal{G}_\varphi}$ associated to an alternative sparse operator by construction. All edge weights are 1.

and $\mathbf{\Psi} = (\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_{15})$ the GFT matrix of $\mathbf{L}$ with basis functions in increasing order of eigenvalues. In particular, we have

$$\mathbf{Z}^{(2)} = \mathbf{\Psi} \cdot \mathrm{diag}(\boldsymbol{\lambda}^{(2)}) \cdot \mathbf{\Psi}^\top,$$
$$\boldsymbol{\lambda}^{(2)} = (0, 0, 2, 2, 0, 0, 0, 2, 2, 0, 0, 2, 2, 0, 0)^\top.$$

Since $\mathbf{Z}^{(2)}$ has only two distinct eigenvalues with high multiplicities, every polynomial of $\mathbf{Z}^{(2)}$ also has two distinct eigenvalues only, which poses a limitation for graph filter design. However, an MPGF with both $\mathbf{Z}^{(1)}$ and $\mathbf{Z}^{(2)}$ still provides more degrees of freedom compared to a PGF with a single operator.

REFERENCES

[1] A. Sandryhaila and J.M.F. Moura, "Discrete signal processing on graphs," *Signal Processing, IEEE Trans. on*, vol. 61, no. 7, pp. 1644–1656, Apr. 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," *Signal Processing Magazine, IEEE*, vol. 30, no. 3, pp. 83–98, May 2013.

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

[4] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovačević, "Signal denoising on graphs via graph filtering," in *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2014, pp. 872–876.

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

[6] A. C. Yağan and M. T. Özgen, "A spectral graph wiener filter in graph fourier domain for improved image denoising," in *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2016, pp. 450–454.

[7] J. Ma, W. Huang, S. Segarra, and A. Ribeiro, "Diffusion filtering of graph signals and its use in recommendation systems," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4563–4567.

[8] N. Tremblay, G. Puy, R. Gribonval, and P. Vandergheynst, "Compressive spectral clustering," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. 2016, ICML16, p. 10021011, JMLR.org.

[9] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *arXiv:1609.02907 [cs, stat]*, Feb. 2017, arXiv: 1609.02907.

[10] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2016, NIPS16, p. 38443852, Curran Associates Inc.

[11] K.-S. Lu and A. Ortega, "Fast graph Fourier transforms based on graph symmetry and bipartition," *IEEE Trans. on Signal Processing*, vol. 67, no. 18, pp. 4855–4869, 2019.

[12] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Autoregressive moving average graph filtering," *IEEE Trans. on Signal Processing*, vol. 65, no. 2, pp. 274–288, Jan 2017.

[13] M. Coutino, E. Isufi, and G. Leus, "Advances in distributed graph filtering," *IEEE Trans. on Signal Processing*, vol. 67, no. 9, pp. 2320–2333, May 2019.

[14] N. Tremblay, P. Gonçalves, and P. Borgnat, "Design of graph filter and filterbanks," in *Cooperative and Graph Signal Processing*, pp. 299–324. Academic Press, June 2018.

[15] A. Sandryhaila and J.M.F. Moura, "Discrete signal processing on graphs: Frequency analysis," *Signal Processing, IEEE Trans. on*, vol. 62, no. 12, pp. 3042–3054, June 2014.

[16] A. Ortega, *Introduction to Graph Signal Processing*, Cambridge University Press, 2021.

[17] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[18] D. I. Shuman, P. Vandergheynst, D. Kressner, and P. Frossard, "Distributed signal processing via Chebyshev polynomial approximation," *IEEE Trans. on Signal and Information Processing over Networks*, vol. 4, no. 4, pp. 736–751, Dec 2018.

[19] S. Segarra, A. G. Marques, and A. Ribeiro, "Optimal graph-filter design and applications to distributed linear network operators," *IEEE Trans. on Signal Processing*, vol. 65, no. 15, pp. 4117–4131, Aug 2017.

[20] A. Loukas, A. Simonetto, and G. Leus, "Distributed autoregressive moving average graph filters," *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1931–1935, 2015.

[21] J. Liu, E. Isufi, and G. Leus, "Filter design for autoregressive moving average graph filters," *IEEE Trans. on Signal and Information Processing over Networks*, vol. 5, no. 1, pp. 47–60, July 2019.

[22] A. Gavili and X. Zhang, "On the shift operator, graph frequency, and optimal filtering in graph signal processing," *IEEE Trans. on Signal Processing*, vol. 65, no. 23, pp. 6303–6318, Dec 2017.

[23] K.-S. Lu, A. Ortega, D. Mukherjee, and Y. Chen, "Efficient rate-distortion approximation and transform type selection using Laplacian operators," in *2018 Picture Coding Symposium (PCS)*, June 2018, pp. 76–80.

[24] N. Emirov, C. Cheng, J. Jiang, and Q. Sun, "Polynomial graph filter of multiple shifts and distributed implementation of inverse filtering," *arXiv:2003.11152*, March 2020.

[25] H. Ochoa-Domínguez and K.R. Rao, *Discrete Cosine Transform*, CRC Press, 2nd edition, 2019.

[26] G. Strang, "The discrete cosine transform," *SIAM review*, vol. 41, no. 1, pp. 135–147, 1999.

[27] M. Püschel and J. M. F Moura, "The algebraic approach to the discrete cosine and sine transforms and their fast algorithms," *SIAM Journal on Computing*, vol. 32, no. 5, pp. 1280–1316, 2003.

[28] Y. Park and H. Park, "Design and analysis of an image resizing filter in the block-DCT domain," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 14, no. 2, pp. 274–279, 2004.

[29] H. S. Shin, C. Lee, and M. Lee, "Ideal filtering approach on DCT domain for biomedical signals: index blocked dct filtering method (ib-dctfm)," *J. Med. Syst.*, vol. 34, no. 2, pp. 741–753, Aug. 2010.

[30] U. Tuna, S. Peltonen, and U. Ruotsalainen, "Gap-filling for the high-resolution pet sinograms with a dedicated DCT-domain filter," *IEEE Trans. on Medical Imaging*, vol. 29, no. 3, pp. 830–839, 2010.

[31] C. Zhang, D. Florêncio, and P. A. Chou, "Graph signal processing-A probabilistic framework," *Technical Report*, Apr 2015.

[32] W. H. Chen and S. C. Fralick, "Image enhancement using cosine transform filtering," in *Image Sci. Math. Symp.*, Nov 1976.

[33] B. Chitprasert and K. R. Rao, "Discrete cosine transform filtering," *Signal Processing*, vol. 19, no. 3, pp. 233–245, 1990.

[34] S. A. Martucci, "Symmetric convolution and the discrete sine and cosine transforms," *IEEE Trans. on Signal Processing*, vol. 42, no. 5, pp. 1038–1051, 1994.

[35] C.-C. Tseng and S.-L. Lee, "Minimax design of graph filter using Chebyshev polynomial approximation," *IEEE Trans. on Circuits and Systems II: Express Briefs*, vol. 68, no. 5, pp. 1630–1634, 2021.

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

[37] G. Fracastoro, D. Thanou, and P. Frossard, "Graph transform optimization with application to image compression," *IEEE Trans. on Image Processing*, vol. 29, pp. 419–432, 2020.

[38] J. Fan, C. Tepedelenlioglu, and A. Spanias, "Graph filtering with multiple shift matrices," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3557–3561.

[39] Z. Wang and B.R. Hunt, "The discrete W transform," *Applied Mathematics and Computation*, vol. 16, no. 1, pp. 19 – 48, 1985.

[40] J. Han, A. Saxena, V. Melkote, and K. Rose, "Jointly optimized spatial prediction and block transform for video and image coding," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1874–1884, Apr 2012.

[41] W. Hu, G. Cheung, and A. Ortega, "Intra-prediction and generalized graph Fourier transform for image coding," *Signal Processing Letters, IEEE*, vol. 22, no. 11, pp. 1913–1917, Nov. 2015.

[42] J. Han, Y. Xu, and D. Mukherjee, "A butterfly structured design of the hybrid transform coding scheme," in *Picture Coding Symposium*, 2013, pp. 1–4.

[43] Y. Chen, D. Mukherjee, J. Han, A. Grange, Y. Xu, S. Parker, C. Chen, H. Su, U. Joshi, C.-H. Chiang, and et al., "An overview of coding tools in av1: the first video codec from the alliance for open media," *APSIPA Trans. on Signal and Information Processing*, vol. 9, pp. e6, 2020.

[44] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *Annals of Statistics*, vol. 36, no. 3, pp. 1171–1220, 2008.

[45] H. Kitajima, "A symmetric cosine transform," *IEEE Trans. on Computers*, vol. C-29, no. 4, pp. 317–323, Apr. 1980.

[46] H. Hou, "A fast recursive algorithm for computing the discrete cosine transform," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 35, no. 10, pp. 1455–1461, 1987.

[47] V. Sanchez, P. Garcia, A. M. Peinado, J. C. Segura, and A. J. Rubio, "Diagonalizing properties of the discrete cosine transforms," *IEEE Trans. on Signal Processing*, vol. 43, no. 11, pp. 2631–2641, 1995.

[48] H. Zhang and F. Ding, "On the Kronecker products and their applications," *Journal of Applied Mathematics*, vol. 2013, 06 2013.

[49] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall Press, USA, 3rd edition, 2009.

[50] S. Ikonin, V. Stepin, R. Chernyak, and J. Chen, "Hadamard transform domain filter for video coding," in *Applications of Digital Image Processing XLII*, Andrew G. Tescher and Touradj Ebrahimi, Eds. International Society for Optics and Photonics, 2019, vol. 11137, pp. 259–264, SPIE.

[51] B. Mohar, "The Laplacian spectrum of graphs," in *Graph Theory, Combinatorics, and Applications*. 1991, pp. 871–898, Wiley.

[52] B. Pasdeloup, V. Gripon, G. Mercier, D. Pastor, and M. G. Rabbat, "Characterization and inference of graph diffusion processes from observations of stationary signals," *IEEE Trans. on Signal and Information Processing over Networks*, vol. 4, no. 3, pp. 481–496, Sep. 2018.

[53] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proceedings of the 27th Annual Asilomar Conference on Signals, Systems, and Computers*, 1993, pp. 40–44.

[54] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society (Series B)*, vol. 58, pp. 267–288, 1996.

[55] J. McClellan, T. Parks, and L. Rabiner, "A computer program for designing optimum FIR linear phase digital filters," *IEEE Trans. on Audio and Electroacoustics*, vol. 21, no. 6, pp. 506–526, 1973.

[56] K.-S. Lu, "Sparse DTT operators," [online] https://github.com/kslu/sparseDttOperators.

[57] M. Ceberio and V. Kreinovich, "Greedy algorithms for optimizing multivariate Horner schemes," *ACM SIGSAM Bulletin*, vol. 38, 11 2003.

[58] O. Teke and P. P. Vaidyanathan, "Extending classical multirate signal processing theory to graphs-Part II: M-channel filter banks," *IEEE Trans. on Signal Processing*, vol. 65, no. 2, pp. 423–437, Jan 2017.

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

[60] W.-H. Chen, C. Smith, and S. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. on Communications*, vol. 25, no. 9, pp. 1004–1009, 1977.

[61] A. Smola and R. Kondor, "Kernels and regularization on graphs," in *Learning Theory and Kernel Machines*, Jan 2003, vol. 2777, pp. 144–158.

[62] V. K. Goyal, "Theoretical foundation of transform coding," *IEEE Signal Processing Magazine*, pp. 9–21, Sept 2001.

[63] J. Han, C.-H. Chiang, and Y. Xu, "A level-map approach to transform coefficient coding," in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 3245–3249.

[64] H. Su, M. Chen, A. Bokov, D. Mukherjee, Y. Wang, and Y. Chen, "Machine learning accelerated transform search for AV1," in *2019 Picture Coding Symposium (PCS)*, 2019, pp. 1–5.

[65] B. Li, J. Han, and Y. Xu, "Fast transform type selection using conditional Laplacian distribution based rate estimation," in *Applications of Digital Image Processing XLIII*, Andrew G. Tescher and Touradj Ebrahimi, Eds.

International Society for Optics and Photonics, 2020, vol. 11510, pp. 461–468, SPIE.

[66] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Trans. on Signal and Information Processing over Networks*, vol. 3, no. 3, pp. 467–483, Sep. 2017.

[67] Y. De Castro, T. Espinasse, and P. Rochet, "Reconstructing undirected graphs from eigenspaces," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 16791702, Jan. 2017.

[68] D. Avis and F. Fukuda, "A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra," *Discrete & Computational Geometry*, vol. 8, pp. 295–313, Sep 1992.

[69] J.-Y. Kao, A. Ortega, and S. S. Narayanan, "Graph-based approach for motion capture data representation and analysis," in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct 2014, pp. 2061–2065.