Graph Learning & Fast Transform Coding of 3D River Data

Weihang Liao*, Gene Cheung*, Shogo Muramatsu†, Hiroyasu Yasuda† and Kiyoshi Hayasaka†
* National Institute of Informatics, Tokyo, Japan
† Niigata University, Niigata, Japan

***Abstract*—Collection of physical river measurements across space and time is important towards analysis and prediction of dynamic river flow, and thus early warning and prevention of flood disaster. In this paper, we focus on lossy compression of 3D river data at high quality using predictive graph-based transforms. Specifically, we first divide 3-dimensional river data into non-overlapping temporal frame groups. Data in a frame group t is then predicted using frame group $t-1$, assuming strong temporal correlation. Then for each block in a frame in group t, we learn a sparse inverse covariance matrix from a spatial neighborhood of blocks in the previous frame group via a graphical lasso algorithm with structural constraints. The learned matrix is then interpreted as a graph Laplacian, and graph lifting transform (GLT) or fast graph Fourier transform (FGFT) are employed to encode the prediction residuals efficiently. Experimental results show coding performance gain over conventional DCT and competing graph transform schemes without graph learning.**

I. Introduction

The drastic cost reduction and ubiquity of sensors in the physical environment mean that large volume of data are now easily collected for a wide range of cyber-physical systems, from video surveillance, hyperspectral imaging for crop monitoring, and virtual reality environments with 3D visual data like point clouds. These high-dimensional sensed data are often dense in sampling (thus exhibits large inter-sample correlation) and very large in size. Thus design of suitable coding algorithms that exploit inherent correlation for compression gain while remain computationally practical is important. In this paper, leveraging on recent advances in graph signal processing (GSP) [1], [2], we focus on graph-based compression of 3D volumetric data, in particular, 3D river data.

Climate change in recent decades has led to more frequent and unpredictable torrential rain in Japan, often resulting in flood disasters, property damages and loss of human lives. As a preventive measure, [3] have proposed a large-scale river monitoring and analysis system, so that early warnings for potential flooding can be automatically generated. Dense measurements of river data (*e.g.*, water level, water depth, water speed) across space and time result in large 3D volumetric data, however, which requires compression for efficient network transmission and storage.

In this paper, we propose a graph-based coding framework for compression of large 3D volumetric data. Specifically, we first segment the captured 3D data into groups of finite duration in time, where each group is composed of "frames" of 2D samples across 2D space. For each group t, we first perform prediction using encoded signal in group $t-1$ as predictor to reduce signal energy, assuming strong temporal correlation. To efficiently code the prediction residual, we first perform structure-constrained graph learning [4] using samples in group $t-1$, which efficiently estimates a sparse inverse-covariance (precision) matrix. Given the computed precision matrix, we compress blocks in frames of group t using either *graph lifting transform* (GLT) [5] or *fast graph Fourier transform* (FGFT) [6], the earlier of which has lower complexity—$O(n \log n)$ instead of $O(n^2)$—but slightly lower compression performance. Experimental results show that our coding scheme outperforms conventional DCT and graph transform schemes without graph learning.

The rest of the paper is organized as follows. We first overview related work in Section II. We present the overview of the proposed coding framework in Section III, then describe the details of graph learning and graph based coding in Section IV. Experimental results and conclusion are presented in Section V and VI, respectively.

II. Related Work

Graph transform coding of image and video data has been studied for close to a decade now [5], [7]–[12]. While earlier works focused on coding of piecewise smooth images like depth maps [7]–[9], more recent works have been extended to higher-dimensional visual data like hyperspectral images [11] and light field images [12]. Optimal graph transforms for coding of prediction residuals have also been studied [10], as well as fast implementation of graph transform via lifting [5]. Leveraging on these previous works, in this paper we design a compression scheme for graph transform coding of 3D river data, exploiting inherent inter-sample correlation in the data via graph learning [4], and utilizing recent advances on fast implementation such as FGFT [6].

III. Overview of the Coding Scheme

We first define notations used in this paper. We then describe how we organize the 3D-volumetric data, and then overview our proposed coding framework.

Given a generic 3D-volumetric data, we assume that the samples are regularly distributed along x and y dimensions in 2D space and across time t. At each sample location, there are n channels. One typical example is video, where each pixel contains R, G and B color channels. In order to elucidate the

sediment transport mechanism of rivers, we developed Stream Tomography, a device capable of simultaneously measuring the variations of surface and bed of an artificial indoor river, and acquire them as time series data. For the river data, it contains four channels: the height of the river surface, the depth of the river bed, the variation of the river height, and the variation of the river depth.

To be adaptive to non-stationary statistics across time, we first divide the whole 3D-volumetric data into K frame groups along the time dimension. In each group, the data is organized into *frames*, each containing samples of different spatial locations but the same time instant. The k-th group is denoted by $\mathbf{g}_k$, each group contains M frames, and the m-th frame in $\mathbf{g}_k$ is denoted by $\mathbf{f}_k^m$. In each frame, the data is divided into non-overlapping blocks, and the locations of different blocks are denoted by a pair of coordinates (x, y). The data organization is illustrated in Fig. 1.

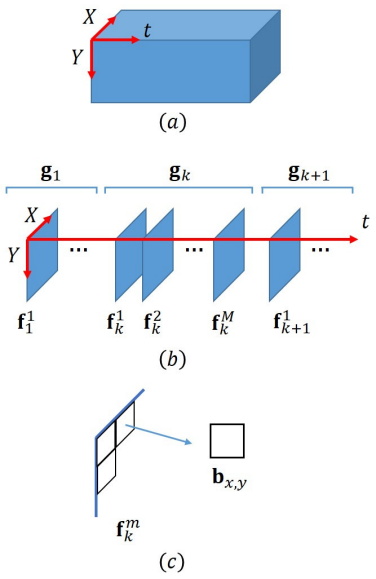

Fig. 1. (a)the original 3D-volumetric data; (b)the partition of different groups and frames; (c)the blocks in one frame.

We assume a *Gaussian Markov random field* (GMRF) model for each sample block in a frame in a group, whose statistics are similar to a neighborhood of blocks in frames of a previous group. We assume further that four channels within the same block are correlated, and hence once one channel is encoded, we can estimate its inter-sample correlations for coding of the other channels.

For the first channel in group $\mathbf{g}_k$, we use the last frame in the previous group $\mathbf{f}_{k-1}^M$ as an reference frame to predict the other frames $\mathbf{f}_{k-1}^m$, $m \in \{1, \ldots, M-1\}$. Given the computed prediction residuals, we then use a graph learning algorithm [4] to learn a sparse inverse covariance (precision) matrix to capture inter-pixel correlations. For the remaining three channels, we instead use the decoded first channel in the same group $\mathbf{g}_k$ to estimate the precision matrix, with the observation that the channels in same time instant have stronger correlations.

Once a sparse precision matrix is computed, it is represented as a graph, and we use different implementations of *Graph Fourier Transform* (GFT) [1], [9] to compute transform coefficients, perform quantization and entropy coding using the Amplitude and Group Partition (AGP) method [13]. For fast implementation of GFT, we employ two variants for practical execution: Graph Lifting Transform (GLT) [14] and Fast Graph Fourier Transform (FGFT) [6].

At the decoder side, we follow the same procedure described above to compute a prediction residual, and learn exactly the same graph, thus arriving at the same transform basis. Notice that in our coding scheme, no additional side information (SI) is required that would result in extra coding overhead, unlike [9].

IV. Graph Based Coding

We first discuss how we construct a graph structure for coding of a sample block, then introduce the two GFT approximations—GLT and FGFT—respectively.

A. Graph Definition

The proposed coding scheme is executed block-by-block for each frame. For one block we first construct an undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$. The vertex set $\mathcal{V}$ corresponds to the data sample locations in the block, and the edge set $\mathcal{E}$ is specified by $(i, j, w_{i,j})$, where $i, j \in \mathcal{V}$, and $w_{i,j} \in \mathbb{R}^+$ is the positive edge weight between 0 and 1 that reflects the similarity between connected vertices i and j. Adjacency matrix $\mathbf{W}$ is a $|\mathcal{V}| \times |\mathcal{V}|$ symmetric matrix, where the (i, j)-th element $W_{i,j}$ is $w_{i,j}$. We then define the degree matrix $\mathbf{D}$ as a $|\mathcal{V}| \times |\mathcal{V}|$ diagonal matrix, where diagonal element $D_{i,i} = \sum_j W_{i,j}$. Finally, the graph Laplacian matrix of $\mathcal{G}$ is defined as: $\mathbf{L} = \mathbf{D} - \mathbf{W}$ [1].

B. Graph Learning

As discussed previously, using a training dataset (either blocks of the same channel in similar spatial locations in the previous group or blocks of different channels in the same group) we compute a sparse inverse covariance matrix using algorithm in [4], which we interpret as the graph Laplacian matrix $\mathbf{L}$. The algorithm [4] is basically a graphical lasso algorithm [15] with structural constraints; it estimates the target Laplacian matrix $\mathbf{L}$ by minimizing the following objective:

$$\min_{\mathbf{L}} \ \mathrm{Tr}(\mathbf{LC}) - \log\det(\mathbf{L}) + \|\mathbf{L} \odot \mathbf{H}\|_1 \qquad (1)$$

where $\mathbf{C}$ is an empirical covariance matrix, and $\mathbf{H}$ is a regularization matrix. Since $\mathbf{L}$ is singular by construction, the objective function in (1) is rewritten as:

$$\begin{aligned} &\min_{\mathbf{L}} \ \mathrm{Tr}(\mathbf{L}(\mathbf{C} + \mathbf{H} + \mathbf{J})) - \log\det(\mathbf{L} + \mathbf{J}) \\ &\text{s.t. } \mathbf{L} \in \mathcal{L}(\mathbf{A}) \end{aligned} \qquad (2)$$

where $\mathbf{J} = \mathbf{v}\mathbf{v}^T$ ($\mathbf{v}$ is a constant vector of length $|\mathcal{V}|$). $\mathbf{J}$ ensures that the determinant is larger than 0. $\mathbf{A}$ is a

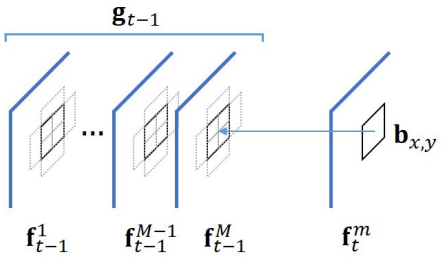

Fig. 2. The shifting of reference block. The blocks with black solid line are in the same spatial location, the blocks with gray dashed line are the shifted locations.

connectivity matrix specifying the permissible locations of non-zero edge weights. $\mathcal{L}(\mathbf{A})$ is a set of graph Laplacians which follow the structure constraints specified by $\mathbf{A}$.

In our coding scheme, we constrain $\mathcal{L}(\mathbf{A})$ to be the space of 8-connected graphs to capture only neighboring sample correlations. The optimization in (2) is solved by an iterative block-coordinate descent algorithm [16].

Note that there are roughly $4 \times |\mathcal{V}|$ edges in the graph. To collect sufficient observations for graph learning, when constructing a graph for a block at coordinate (x, y) in the first channel in group t, we consider the corresponding (overlapping) blocks at coordinate (x', y') in a spatial neighborhood $\pm\rho$ in all frames in the previous group $t-1$, as shown in Fig. 2. We can thus collect $N = (M-1) \times (2\rho+1)^2$ observations to compute an empirical covariance matrix $\mathbf{C}$.

C. Graph Spectrum

Once the graph Laplacian matrix has been learned, we can perform eigen-decomposition:

$$\mathbf{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\top} \tag{3}$$

where $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues of $\mathbf{L}$ along its diagonal, and $\mathbf{V}$ contains eigenvectors of $\mathbf{L}$ as columns. Together they form the *graph spectrum* of the graph $\mathcal{G}$. Note that $\mathbf{L}$ is symmetric and thus all its eigenvalues are real, and $\mathbf{L}$ can be shown to be positive semi-definite (PSD)—via the Gershgorin circle theorem—and thus all its eigenvalues are also non-negative. $\mathbf{V}^{\top}$ is also known as the *graph Fourier Transform* (GFT).

To compute the GFT, however, the complexity of general eigen-decomposition of a $n \times n$ matrix is $O(n^3)$. To reduce the computation cost, we investigate two fast implementations in the following subsections.

D. Graph Lifting Transform

The GLT is implemented based on the idea of wavelet transform [14], namely, use lifting to iteratively split the graph into approximate and detailed subgraphs. The lifting operation contains three main steps:

1) *Bipartition*. The input graph is split into two different subsets: the prediction set $\mathcal{P}$ and the update set $\mathcal{U}$, where $\mathcal{P} + \mathcal{U} = \mathcal{V}$ and $\mathcal{P} \cup \mathcal{U} = \varnothing$.
2) *Prediction stage*. Every node in $\mathcal{P}$ is predicted by some neighbors in $\mathcal{U}$ with a prediction filter $\mathbf{p}$. The prediction residual $\mathbf{d}$ corresponds to the detail coefficients.
3) *Update stage*. Every node in $\mathcal{U}$ is updated with $\mathbf{d}$ using the update filter $\mathbf{u}$. This results in smooth coefficients $\mathbf{s}$, which is an approximation of the original signal.

After one lifting operation, the output smooth coefficients $\mathbf{s}$ will be input to next iteration, resulting in a multi-level decomposition. The GLT has been shown to achieve an approximate performance to GFT [17], while no eigen-decomposition is required, and the cost of computing the GLT coefficients is only $O(n \log n)$.

In our implementation, we solve the bipartition problem based on the idea in [18]. We first construct a *minimum spanning tree* of the input graph $\mathcal{G}$, then use *breadth first search* to traverse every nodes. So the vertex set $\mathcal{V}$ will be split into $\mathcal{P}$ and $\mathcal{U}$ based on the odd-even depth. The $\mathbf{p}$ and $\mathbf{u}$ filters are designed based on the method proposed in [19], where the prediction filter $\mathbf{p}$ is optimized to minimize the energy of the detail coefficients $\mathbf{d}$, and the update filter $\mathbf{u}$ is designed to be orthogonal to $\mathbf{p}$. The transform coefficients are uniformly quantized and reordered based on [20].

E. Fast Graph Fourier Transform

The motivation of FGFT is to avoid expensive eigen-decomposition of the graph Laplacian $\mathbf{L}$, while approximating the eigenvectors in $\mathbf{V}^{\top}$. In our coding scheme, FGFT is implemented based on the method proposed in [6], *i.e.*, a series of sparse and orthogonal matrices (rotation matrices called *Givens matrices*) to approximately diagonalize the graph Laplacian matrix $\mathbf{L}$:

$$\mathbf{L} \approx \mathbf{S}_1 ... \mathbf{S}_J \hat{\Lambda} \mathbf{S}_J^T .. \mathbf{S}_1^T \tag{4}$$

where $\hat{\Lambda}$ is an approximately diagonal matrix, and its diagonal elements are approximations of eigenvalue λ. The product $\hat{\mathbf{V}} = \mathbf{S}_1 ... \mathbf{S}_J$ form the approximated transform basis.

In order to find a good approximation with reasonable complexity, the author used a truncated Jacobi eigenvalue algorithm [21]. It is an iterative procedure, where at each step it finds the Givens rotation matrix $\mathbf{S}_j$ that eliminates the largest remaining entry in $\mathbf{L}$. See Fig. 2 and Fig. 3 in [6] for more details.

To decide when to appropriately terminate the Jacobi eigenvalue algorithm (resulting in an approximately diagonalized $\mathbf{L}$), we set the maximum number of iterations J with respect to the size of the graph:

$$J = 6|\mathcal{V}| \cdot \lfloor \log_2(|\mathcal{V}|) \rfloor \tag{5}$$

Note that FGFT only avoids the eigen-decomposition (Eqn. 3). To compute transform coefficients, we still need to perform matrix-vector multiplication, which is $O(n^2)$.

V. Experiments

A. Experiment Data

In order to validate the efficiency of our proposed 3D data coding scheme, we collected river depth data from an artificial

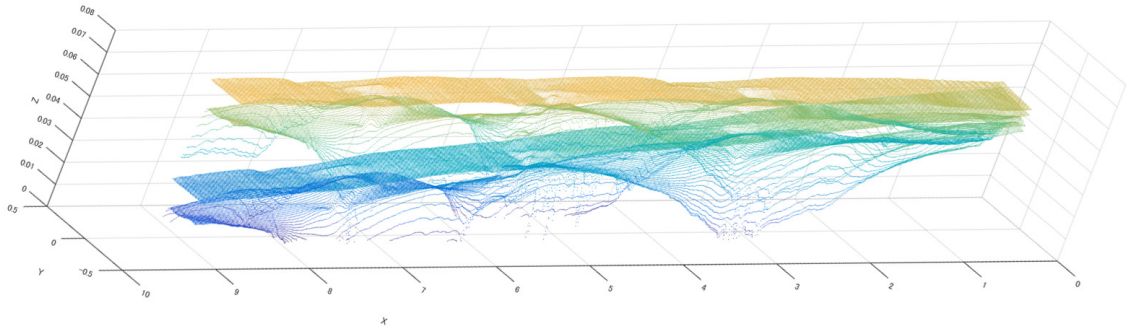

Fig. 3. One time instant of the 3D river data, The x axis is the length-wise spatial displacement (down the river), and the y axis is the width-wise spatial displacement (across the river). Different colors represent different depths for visualization (blue corresponds to deep, yellow corresponds to shallow).

indoor river testbed in Niigata University, by using a stream tomography sensing device. The dimension of the monitored river is of 12m length and 0.45m width, and the gradient of the bed slope is 1/200. The data is sampled on a regular 2D grid with dimension 41×851, and at each sample location there are 4 channels recorded (the height of the river surface, the depth of the river bed, the variation of the height, and the variation of the depth). By analyzing the data we notice that channels 1 and 3 are correlated, and channels 2 and 4 are correlated. So for the experiment we use channel 1 to predict 3, and channel 2 to predict 4, respectively. The samples are collected periodically at 10 minute intervals, and 44 time instants in total. One time instant of the 3D river data is shown in Fig. 3.

B. Experiment Settings

We first organize the 3D river data into frame groups, as discussed in Section. III. For each frame we use 8×8 block as the basic coding unit. The *QP* values are from 4 to 24, with step size 4. Based on these configurations we design several comparison tests, as described below:

1) We implement the entire proposed coding scheme, and test both the low complexity (GLT) and high complexity (FGFT) implementations separately.
2) We test a coding scheme without graph learning algorithm. GLT is employed on a 8-connected graph constructed based on Euclidean distance of sample locations.
3) We test the coding performance of standard Discrete Cosine Transform (DCT) directly applied on the prediction residual.

C. Performance Comparison

The coding performances of different coding schemes in two sets of experiments are shown in Fig. 4 and 5. For both small and large group size, we can observe that the proposed coding scheme yields obvious coding gain, especially for the high PSNR region. Specifically, at PSNR=$55dB$, our coding scheme using FGFT can save 60% bits consumption compare with the default DCT; While the coding scheme using GLT can save 40% bits consumption. Notice that for this application of river data storage and analysis, the monitored river data must be stored in a very high quality, thus the coding gain in high PSNR region is meaningful.

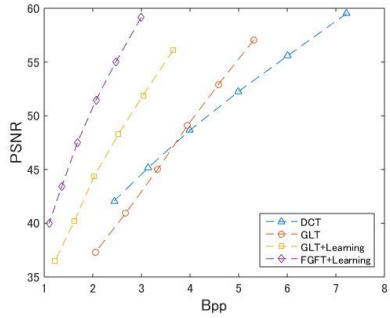

Fig. 4. The coding performance comparison, each group contains 4 frames.

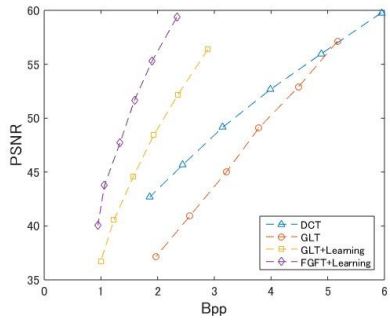

Fig. 5. The coding performance comparison, each group contains 11 frames.

VI. Conclusions

Compression of 3D river data is important for early prevention and warning of river flooding due to torrential rainfalls. In this paper, leveraging on recent advance in graph-based transform coding, we design a new compression scheme for large 3D river data. Experimental results show that by learning appropriate graph structures from previously coded data, one can utilize fast implementation of graph transforms for practical and efficient compression of river data.

REFERENCES

[1] 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," in *IEEE Signal Processing Magazine*, vol. 30, no.3, May 2013, pp. 83–98.

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

[3] "Environmental monitoring applications: Flood warning systems," https://www.fondriest.com/environmental-measurements/environmental-monitoring-applications/flood-warning-systems/, accessed Jun 2, 2018.

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

[5] Y.-H. Chao, A. Ortega, W. Hu, and G. Cheung, "Edge-adaptive depth map coding with lifting transform on graphs," in *31st Picture Coding Symposium*, Cairns, Australia, May 2015.

[6] L. Le Magoarou, R. Gribonval, and N. Tremblay, "Approximate fast graph fourier transforms via multilayer sparse approximations," *IEEE transactions on Signal and Information Processing over Networks*, vol. 4, no. 2, pp. 407–420, 2018.

[7] G. Shen, W.-S. Kim, S. Narang, A. Ortega, J. Lee, and H. Wey, "Edge-adaptive transforms for efficient depth map coding," in *IEEE Picture Coding Symposium*, Nagoya, Japan, December 2010.

[8] W. Hu, G. Cheung, X. Li, and O. Au, "Depth map compression using multi-resolution graph-based transform for depth-image-based rendering," in *IEEE International Conference on Image Processing*, Orlando, FL, September 2012.

[9] W. Hu, G. Cheung, A. Ortega, and O. Au, "Multi-resolution graph Fourier transform for compression of piecewise smooth images," in *IEEE Transactions on Image Processing*, vol. 24, no.1, January 2015, pp. 419–433.

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

[11] J. Zeng, G. Cheung, Y.-H. Chao, I. Blanes, J. Serra-Sagrista, and A. Ortega, "Hyperspectral image coding using graph wavelets," in *IEEE International Conference on Image Processing*, Beijing, China, September 2017.

[12] Y.-H. Chao, G. Cheung, and A. Ortega, "Pre-demosiac light field compression using graph lifting transform," in *IEEE International Conference on Image Processing*, Beijing, China, September 2017.

[13] A. Said and W. A. Pearlman, "Low-complexity waveform coding via alphabet and sample-set partitioning," in *Visual Communications and Image Processing'97*, vol. 3024. International Society for Optics and Photonics, 1997, pp. 25–38.

[14] S. Narang and A. Ortega, "Lifting based wavelet transforms on graphs," in *APSIPA ASC*, Sapporo, Japan, October 2009.

[15] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," in *Biostatistics*, vol. 9, no.3, 2008, pp. 432–441.

[16] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

[17] Y.-H. Chao, A. Ortega, and S. Yea, "Graph-based lifting transform for intra-predicted video coding," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1140–1144.

[18] G. Shen and A. Ortega, "Optimized distributed 2d transforms for irregularly sampled sensor network grids using wavelet lifting," in *IEEE International Conference on Acoustics, Speech and Signal Processing, 2008.* IEEE, 2008, pp. 2513–2516.

[19] E. Martinez-Enriquez, J. Cid-Sueiro, F. Diaz-De-Maria, and A. Ortega, "Directional transforms for video coding based on lifting on graphs," *IEEE Transactions on Circuits and Systems for Video Technology*, 2016.

[20] E. Martínez-Enríquez, F. Díaz-de María, and A. Ortega, "Video encoder based on lifting transforms on graphs," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*. IEEE, 2011, pp. 3509–3512.

[21] G. H. Golub and H. A. Van der Vorst, "Eigenvalue computation in the 20th century," *Journal of Computational and Applied Mathematics*, vol. 123, no. 1, pp. 35–65, 2000.