Critically Sampled Graph-based Wavelet Transforms for Image Coding

Sunil K. Narang,* Yung-Hsuan Chao,† and Antonio Ortega‡
† Ming Hsieh Department of Electrical Engineering
University of Southern California
E-mail: kumarsun@usc.edu, yunghsuc@usc.edu, ortega@sipi.usc.edu

***Abstract*—In this paper, we propose a new approach for image compression using graph-based biorthogonal wavelet filterbanks (referred to as graphBior filterbanks). These filterbanks, proposed in our previous work, operate on the graph representations of images, which are formed by linking nearby pixels with each other. The connectivity and the link weights are chosen so as to reflect the geometrical structure of the image. The filtering operations on these *edge-aware* image graphs avoid filtering across the image discontinuities, thus resulting in a significant reduction in the amount of energy in the high frequency bands. This reduces the bit-rate requirements for the wavelet coefficients, but at the cost of sending extra edge-information bits to the decoder. We discuss efficient ways of representing and encoding this edge information. Our experimental results, based on the SPIHT codec, demonstrate that the proposed approach achieves better R-D performance than the standard CDF9/7 filter on piecewise smooth images such as depth maps.**

I. Introduction

The wavelet transform has been successfully used in image compression for many years due to its ability to exploit spatial and frequency correlation in typical images. Image discontinuities, e.g., strong edges in between smooth regions, require a significant amount of rate in typical image coders. This is because the standard separable wavelet transform produces large magnitude coefficients around the edge location across multiple subbands. At low bit-rates, significant errors in representing these coefficients lead to ringing artifacts in the reconstructed images.

In order to alleviate this problem, researchers have made efforts to design directional wavelet filterbanks based on the image geometry in order to emphasize the filtering along edges rather than across edges. In general, previous works can be divided into two categories: designing filters adaptively based on the geometric flows of the images [3][2], or applying fixed filterbanks on the rotated images [11][1]. However, these approaches suffer from two drawbacks. First, the design complexity is high [3][2]. For instance, the design of adaptive filters requires image classification and the filter selection. Second, over-sampling [11][1] may not suitable for image compression. To alleviate the over-sampling problem, a set of critically sampled contourlets is proposed in[4] but this approach involves significant complexity for directional filter design and FIR approximation. Moreover, all of the filters mentioned above are non-separable.

In [7] and [5], graph-based wavelet filterbanks with quadrature mirror filters (QMF) are proposed and applied to images. The framework achieves edge adaptation while using critically sampled separable filterbanks, as in standard wavelet transforms. Edge awareness is achieved by representing images as graphs, with each pixel corresponding to a graph node. The advantage of a graph structure lies in its flexibility for expressing diverse image geometries. In the image compression case, we assign smaller (but fixed) link weights to pairs of pixels that are in different sides of an image edge, and transmit the location of these *weak links* as side information to the decoder. As will be shown, this selection can reduce the amount of energy due to edges present in the high frequency wavelet subbands. In [5] we showed that our method achieves promising results when compared with a standard wavelet approach based on non-linear approximation performance.

In this paper, we use these ideas to build a practical system for image compression. We make use of our proposed biorthogonal wavelet filterbanks (graphBior)[6], due to their perfect reconstruction and compact support properties. In particular, we show that the $graphBior(2,2)$ filterbanks are identical to standard CDF filterbanks for unweighted 4 connected image graphs, and provide a natural extension of standard filters on edge-aware weighted image graphs. Further, we also include, for the first time, the cost of sending extra edge information to the decoder in the overall transform cost. For this, we propose efficient ways for the generation, compression and transmission of edge side information to the decoder. We encode the wavelet coefficients generated by our filters using set partitioning in hierarchical trees (SPIHT)[9]. The paper is organized as follows: In Section II, we will review graph wavelet filterbanks, and their application to image signals. The generation, compression and encoding of edge information is discussed in Section III. We present our results in Section IV, and conclusion in Section V.

II. Graph Wavelet Transforms for Images

A. Graph Representation

In the graph signal processing (GSP) framework [10], we represent images as undirected graphs $G = (\mathcal{V}, E)$, where $\mathcal{V}$ is the set of pixels (nodes) indexed as $1, 2, 3, ..., N$, and E is the set of links between pixels. A link is denoted as a triplet (i, j, w_{ij}), where i and j are the end pixels and w_{ij} is the link strength which is often measured as the spatial and/or photometric similarity between pixels i and j. Note that the graph representation is not unique, and the choice of a particular topology depends upon various factors such as the geometric structure of the image, the desired complexity of the resulting graph filters, and the side information needed to generate identical graphs at both encoder and decoder. In this paper, we choose a 4-connected graph representation of images, as shown in Figure 2(a), in which each pixel is connected to its 4 nearest neighbors in horizontal and vertical directions. In an *unweighted graph representation* all links have equal weights, whereas in an *edge-aware graph representation* the weights are adapted to the given image, as will be discussed in Section III.

B. Graph Signals and Graph Filters

The pixel intensities in GSP framework are represented as a $1D$ vector $\mathbf{x}$, such that $x(i)$ is the value of pixel i. We denote the adjacency matrix as $\mathbf{A}$, where $A(i,j) = w_{ij}$ if $(i, j, w_{ij}) \in E$ and 0 otherwise. Given $\mathbf{A}$, the Laplacian matrix as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ (called the degree matrix) is diagonal, with $D(i,i) = \sum_j A(i,j)$. The corresponding symmetric normalized Laplacian matrix is $\boldsymbol{\mathcal{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$. We consider only undirected graphs without self

loops for which $\boldsymbol{\mathcal{L}}$ is a symmetric positive semi-definite matrix. Therefore, it has the eigenvalue decomposition:

$$\boldsymbol{\mathcal{L}} = \mathbf{U}\Lambda\mathbf{U}^t = \sum_{i=1}^{N} \lambda_i \mathbf{u}_i \mathbf{u}_i^t, \tag{1}$$

with a diagonal eigenvalue matrix $\mathbf{\Lambda}$ containing non-negative eigenvalues $\{\lambda_1, \lambda_2 \dots \lambda_N\}$ arranged in a non-decreasing order at the diagonal, and a unitary matrix U containing corresponding eigenvectors $\mathbf{u}_i$. The eigenvector-eigenvalue pair is used to define Fourier transform for signals [10]. Eigenvalues λ_i are the graph frequencies and eigenvectors serve as the corresponding projection basis. Every graph signal can be represented with basis $\mathbf{U}$ as $\mathbf{x} = \sum_i \tilde{x_i}\mathbf{u}_i$, where $\tilde{x_i}$ is the projection of $\mathbf{x}$ onto $\mathbf{u}_i$. The graph wavelet filters [10] are defined as:

$$\mathbf{H} = \mathbf{U}\tilde{h}(\Lambda)\mathbf{U}^t = \sum_{i=1}^{N} \tilde{h}(\lambda_i)\mathbf{u}_i \mathbf{u}_i^t, \tag{2}$$

where $\tilde{h}(\lambda_i)$ is the *spectral response* or *spectral kernel* of the filter $\mathbf{H}$. Given input signal $\mathbf{x}$, the output signal can be obtained as $\mathbf{y} = \mathbf{H}\mathbf{x}$ in vertex domain, and as $\tilde{y}_i = \tilde{h}(\lambda_i)\tilde{x}_i$ in the graph frequency domain. Thus, the spectral response $\tilde{h}(\lambda)$ can be designed to enhance or attenuate the contribution of input signal at a frequency λ in the output signal.

C. Graph Wavelet Filterbanks

The graph wavelet filterbanks proposed in [6], [7] are implemented as $1D$ filterbanks on bipartite graphs, and as multi-dimensional separable filterbanks on arbitrary graphs via bipartite subgraph decomposition. A bipartite graph has the form $\mathcal{B} = (L, H, E)$, where L and H are the partitions of the vertex set such that all the links in the set E connect nodes of different partitions. The 4-connected image graph G (Figure 2(a)) used in this paper can be decomposed into 2 bipartite subgraphs $\mathcal{B}_1$ and $\mathcal{B}_2$ (Figure 2(b)-(c)), containing all the links in the horizontal and vertical directions, respectively. These subgraphs are bipartite with their partitions shown as nodes of different colors in 2(b)-(c)). In [7], [5], we proposed *graphQMF* filterbanks which are perfect reconstruction and provide an orthogonal decomposition of signals. However, these filters cannot be implemented as FIR filters in the image domain. As an alternative, in [6] we proposed biorthogonal *graphBior* filterbanks which have compact support and satisfy perfect reconstruction conditions. These filterbanks are specified as *graphBior(k, k)* so that the filter lengths of $\mathbf{H}_0$ and $\mathbf{H}_1$ are $4k+1$ and $4k-1$ [1], respectively, in the image domain. These filterbanks are implemented as $2D$ separable two channel filterbanks on images, as shown in Figure 1. The graphBior filterbanks is described in detail in [6].

In image applications, an all constant signal (a DC signal) has a physical interpretation, and the highpass filters should be designed to have zero response to the DC signal. A direct implementation of graphBior filterbanks does not guarantee this DC annihilation property, as the highpass filters are designed to have zero response to eigenvalue 0 of the normalized Laplacian matrix $\boldsymbol{\mathcal{L}}$, which corresponds to eigenvector $\mathbf{D}^{1/2}\mathbf{1}$, which is not constant. This problem is solved by pre-multiplying input signal with $\mathbf{D}^{1/2}$ before the transform and post-multiplying output signal with $\mathbf{D}^{-1/2}$ after the transform. Thus, given a wavelet filter $\mathbf{H}$ of the form (2), we operate on the scaled input $\mathbf{x}_s = \mathbf{D}^{1/2}\mathbf{x}$, and the output $\mathbf{y}_s = \mathbf{H}\mathbf{f}_s$ is scaled back to $\mathbf{y} = \mathbf{D}^{-1/2}\mathbf{y}_s$. In an iterative multilevel decomposition, this step is repeated at each level for all filters.

[1]The k in graphBior(k,k) denotes the number of roots in low pass filter $\mathbf{H}_0$ and high pass filter $\mathbf{H}_1$ at λ = 0 and λ =2 respectively. It leads to polynomial filters involving up to $2k$-hops and $(2k-1)$-hops neighbors in the graph, which are equivalent to $4k+1$ and $4k-1$ pixels for line graphs.

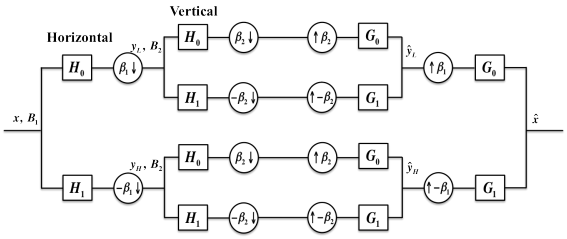


Fig. 1: 2-D separable graph wavelet filterbanks

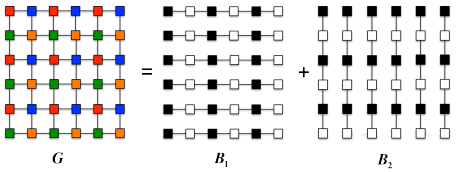


Fig. 2: Two dimension decomposition of images into one bipartite graph with horizontal links and the other with vertical links

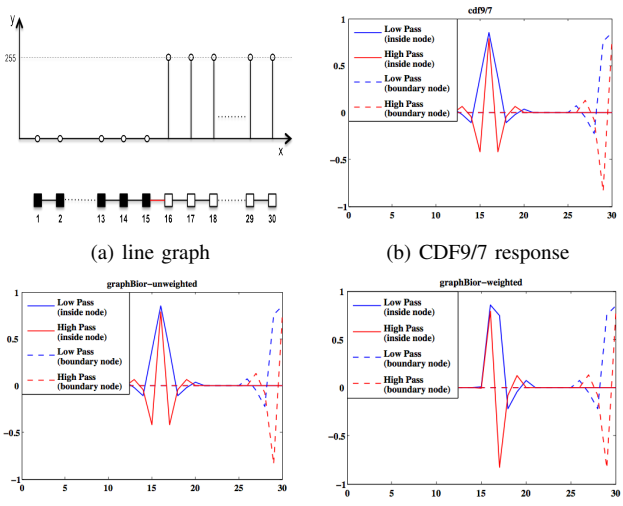
III. Edge-aware Graph Representation

The primary motivation of using edge-aware image graphs in image coding is to assign a lower weight to the links across discontinuities. Figure 3(a) shows a piecewise constant $1D$ image with a sharp discontinuity in the middle. The unweighted $1D$ line graph is formed by linking each pixel with its left and right neighbors with all links having unity weights. **The** $graphBior(2,2)$ **filters on the unweighted line-graph are identical (upto numerical precision** 10^{-12}**) to the standard** $CDF9/7$ **for** $1D$ **signals**. The graph can be made *edge aware* by assigning a lower weight (in this case $w = 0.01$) to the link between pixels 15 and 16, situated at the discontinuity. The impulse responses of the $graphBior(2,2)$ filters at node 16 on the resulting weighted line-graph (Figure 3(d)) are largely concentrated to the right of the node (i.e., away from the discontinuity). This means that the edge-aware graph filters avoid filtering across the edges which leads to reduction in the significant highpass coefficients. This in turn should reduce the bits required to encode wavelet coefficients leading to better compression.

However, in order to use this framework as a practical tool, the following problems must be addressed: a) How to create edge-aware image-graphs (i.e., how to detect weak links in an otherwise regular 4-connected image-graph and assign weights to these weak links)? b) How to efficiently represent and transmit the location and weights of these weak links to the encoder and decoder in order to create identical graphs at both ends. c) The extra edge-information costs extra bits which reduces the gain obtained from reduction in the significant highpass coefficients. How to find an optimal point in this trade-off? In this paper, we provide reasonable but somewhat ad-hoc solutions to each of these problems. A more thorough formulation of these problems is part of our ongoing work.

A. Weak Link Detection

From the graph transform perspective, the optimal way to design edge-aware image graph is to start with an unweighted 4-connected graph, and assign link-weights proportional to some measure of similarity in the intensities of the connected pixels. For example, the link-weights could be computed as bilateral weights as proposed



(a) line graph (b) CDF9/7 response

(c) unweighted graph filter response (d) weighted graph filter response

Fig. 3: impulse response for a 1-D signal in (a) near discontinuity and at the boundary. (a)1-D signal and its line graph representation for graph-based filterbanks. (b) impulse response for CDF9/7 (c) unweighted graph filterbanks. (d) weighted graph filterbanks

in [8]. However, this is not feasible in image coding, since the cost of transmitting the link weights to the decoder would be more expensive than transmitting the raw image itself. The weights can be thresholded, i.e., only links whose weights are below a threshold are assigned non-unity weights. However, it is still expensive to send both weights and location of weak links to the decoder. Besides, thresholding leads to fragmented noisy edges which are not suitable for the transform. Therefore, we break the problem into two parts: First, we apply standard edge detection methods on the input image that avoid the problem of fragmentation and noise, and lead to long connected edges. The output of these edge detection methods is in the pixel domain, i.e., we only obtain pixels situated on the edges (termed as *edge pixels*). In the second step, we find weak links only amongst the links that are directly connected to edge pixels. Further, to minimize the weight information, we assign a lower but fixed weight to all the weak links, which is known to both encoder and decoder. **As a result, we only need to transmit the locations of the weak links to the decoder in order to create identical graphs.**

In the first step, any standard edge-detection method can be used (we use Prewitt's method in our experiments). This works well for piecewise constant images such as that in Figure 5(a). However, natural images are often noisy or have texture regions which create lots of small fragmented edges, and so should be supressed or ignored. A popular method for finding piecewise constant approximation of any image is iterative bilateral filtering [12]. In our experiments, we use bilateral filtering to obtain a smooth piecewise constant image and use it for edge detection.

In the second step, we need to find weak links connected to edge-pixels. A simple way to do this is by thresholding all the links connected to the edge pixels according to their weights. This is problematic for two reasons: First, this method sometimes creates weak links on both sides of an edge pixel in the horizontal or vertical direction. The resulting graph filters in that direction avoid filtering across pixels on both sides. As a result, the edge pixel has no change in its value after filtering in one direction, and this sometimes appear as a discontinuity in the other direction. Second, the weak links obtained by this method are not completely contained on one side of the edge. As a result the graphs of objects separated by the edge are not completely disconnected, and filtering still occurs across the edges which leads to poor performance.

We propose a method that avoids these problems by a) allowing at most one weak link in each direction per edge-pixel, and b) detecting weak links according to the direction of the edge. In our proposed method, we compute intensity gradients $g_h(i)$ and $g_v(i)$ in the horizontal and vertical directions, respectively, and the angle of the gradient $\theta(i) = atan(g_v(i)/g_h(i))$ at each pixel i of the input image. For example, $|\theta(i)| \approx 0$ implies an almost horizontal edge, therefore at such pixels the weak link is detected in the vertical direction only. The vertical link across which the intensity change is maximum is selected as weak link. The other values of $\theta(i)$ are interpreted similarly. The algorithm to detect weak-links is given in Algorithm 1. In a multilevel decomposition, the the link-weights of the downsampled graph are computed as the sum of the weights of the 2-hop paths connecting LL nodes in the previous level graph, where the weight of the path is equal to the product of the weights of the links that it consists of.

Algorithm 1 Algorithm to detect weak links in an image graph. $I(v)$: pixel intensity, $\partial I(v,u) = |I(v) - I(u)|$, $\partial I(v,\mathcal{S}) = \sum_{u\in\mathcal{S}} \partial I(v,u)$

Perform edge detection to obtain edge pixels in the image.
for each edge pixels u **do**
  Compute gradient angle $\theta(u)$.
  if horizontal edge ($|\theta(u)| \approx 0$) **then**
    Define $\mathcal{S}1$ = {topLink}, $\mathcal{S}2$ = {bottomLink};
  else if vertical edge ($|\theta(u)| \approx \pi/2$) **then**
    Define $\mathcal{S}1$ = {leftLink}, $\mathcal{S}2$ = {rightLink};
  else if acute angle edge ($\theta(u) > 0$) **then**
    Define $\mathcal{S}1$ = {topLink, leftLink}, $\mathcal{S}2$= {bottomLink ,rightLink};
  else if obtuse angle edge ($\theta(u) < 0$) **then**
    Define $\mathcal{S}1$ = {topLink, rightLink}, $\mathcal{S}2$= {bottomLink ,left-Link};
  end if
  if $\partial I(u,\mathcal{S}1) < \partial I(u,\mathcal{S}2)$ **then**
    Store links in set $\mathcal{S}2$ as weak links.
  else
    Store links in set $\mathcal{S}1$ as weak links.
  end if
end for

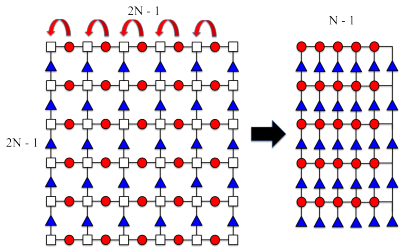


Fig. 4: Graph links representation used for encoding. The $(2N-1)\times(2N-1)$ link map can be shrunk into $(2N-1)\times(N-1)$ map on the right by discarding the nodes corresponding to image pixels since they do not carry the information needed for graph formation

B. Edge Information Encoding

Once we decide the locations of *weak* links, we can form an edgemap as in Fig. 4(left), where the circles and triangles indicate

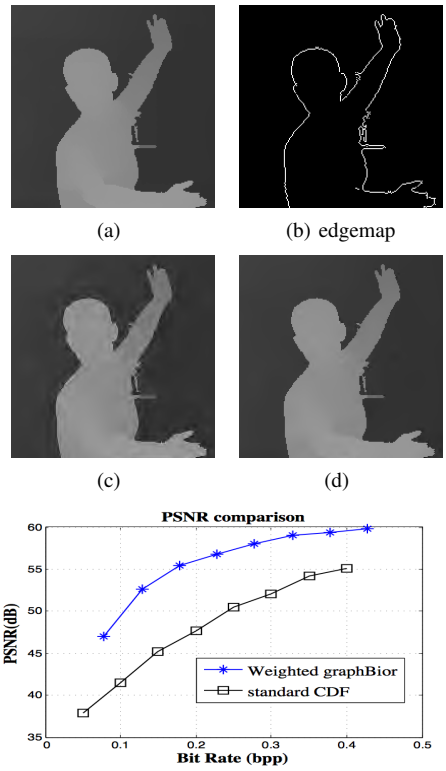
the horizontal and vertical links between image pixels (squares), respectively. We assign 0 to all the *regular* links and 1 to all *weak* links. Since the image pixels contribute nothing to the graph construction, we can remove them from the edgemap to obtain a reduced edgemap as in Fig. 4 (right), which can be stored as a binary image of size $(2N-1)\times(N-1)$. Further, we assign a fixed weight w to all weak links. As a result the reduced edgemap is sufficient to generate identical image graphs at both encoder and decoder. We encode the $(2N-1)\times(N-1)$ binary map by JBIG, which is the binary image coding standard, and include the bits needed to encode JBIG edgemap in the rate-distortion (R-D) evaluation.

IV. Experiments

In this section, we use ballet image as our test image (shown in Fig. 5(a)). The image is represented using a 4-connected graph. We use *Prewitt* edge-detection algorithm to obtain edge-pixels in the image , and then follow the algorithm given in Algorithm 1, to detect weak links. From this we generate the reduced edgemap and encode it as a JBIG image. The transmission of the JBIG file along with the encoded wavelet coefficients requires 0.028 extra bits per pixels (bpp), which is included in the R-D evaluation. We apply 5-level wavelet decomposition and the coefficients, in both the proposed methods and in standard $CDF9/7$ implementation, are encoded using the Set Partitioning In Hierarchical Trees (SPIHT) algorithm [9]. Figure 5 shows the comparison of proposed method with the standard filterbank. The figures (Fig. 5(c) and Fig. 5(d)) show reconstructed images with wavelet coefficients encoded in bit rate = 0.05 bpp. It can be observed that the edge-aware graph filterbanks provide significantly better perceptual quality than the standard filterbanks. Further, the edges in the image are better preserved with proposed method than with the standard filterbanks, which is quite important in depth-map compression applications. The PSNR performance is shown in Figure 5(e), where we consider the bit rate from 0.05bpp to 0.4 bpp with a step size of 0.05. Note that the curve shift for weighted graphBior is due to the cost of edge information. It can be seen that even after including the cost of extra edge information, the edge-aware graphBior filterbanks provide upto $7dB$ gain in PSNR over standard filterbanks. The results of proposed method on standard uncompressed natural images such as lena, barbara, cameraman etc. (all of size 512×512 px), did not provide any gain over standard CDF filters. The reason is that the most of the edges found in these images are already blurry (smooth) at this resolution, and therefore do not produce significant highpass coefficients with standard wavelets.

V. Conclusions

In this paper, we used the ideas from our previous work [5] to design a practical system for image compression. Our proposed method operates on graphs constructed from images, in which links are adjusted to adapt to the edge-structure of the image. In addition, we provided an efficient scheme for encoding edge information to reconstruct the graphs at the decoder side. The results on piecewise constant images show that even with edge information encoded, the graphBior filterbanks provide better compression than standard $CDF9/7$ filterbanks. However, the proposed design in this paper is based on heuristics and does not optimize the tradeoff between the performance gain and cost of transmitting extra edge-information. The formulation of this tradeoff and extension of our approach to natural images are our future work.



(e)

Fig. 5: Reconstructed Ballet using bit-rate = 0.05 of encoded wavelet coefficients. (a) Original Image, (b) Edgemap (c) Standard $CDF9/7$, (d) Edge aware graphBior filters (e) R-D plot: SPIHT encoding

References

[1] M. N Do and M. Vetterli. The contourlet transform: an efficient directional multiresolution image representation. *IEEE Trans. Image Processing*, 14(12):2091–2106, 2005.

[2] D. L. Donoho. Wedgelets: nearly minimax estimation of edges.(english summary). *Ann. Statist*, 27(3):859–897, 1999.

[3] E. L. Pennec and S. Mallat. Sparse geometric image representations with bandelets. *IEEE Trans. Image Proc.*, 14(4):423–438, 2005.

[4] Y. Lu and M. N. Do. Crisp contourlets: a critically sampled directional multiresolution image representation. In *Optical Science and Technology, SPIE's 48th Annual Meeting*, pages 655–665. International Society for Optics and Photonics, 2003.

[5] S. K Narang, Y. H. Chao, and A. Ortega. Graph-wavelet filterbanks for edge-aware image processing. In *Statistical Signal Processing Workshop (SSP)*, pages 141–144. IEEE, 2012.

[6] S. K Narang and A. Ortega. Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs. *arXiv preprint arXiv:1210.8129*, 2012.

[7] S. K Narang and A. Ortega. Perfect reconstruction two-channel wavelet filter banks for graph structured data. *IEEE Trans. Signal Processing*, 60(6):2786–2799, 2012.

[8] S.K. Narang, A. Gadde, and A. Ortega. Signal processing techniques for interpolation of graph structured data. *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, 2013. to appear.

[9] A. Said and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. Circuits and systems for video tech.*, 6(3):243–250, 1996.

[10] D. I Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE sig. proc. mag.*, 30(3):83–98, May 2013.

[11] J. Starck, E. J. Candès, and D. L. Donoho. The curvelet transform for image denoising. *IEEE Trans. Image Processing*, 11(6):670–684, 2002.

[12] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision*, pages 839–846, Jan 1998.