PART 1

Fundamentals of Graph Signal Processing

1

Graph Spectral Filtering

Yuichi TANAKA
Tokyo University of Agriculture and Technology, Japan

1.1. Introduction

The filtering of time- and spatial-domain signals is one of the fundamental techniques for image processing and has been studied extensively to date. GSP can treat signals with irregular structures that are mathematically represented as graphs. Theories and methodologies for the filtering of graph signals are studied using spectral graph theory. In image processing, graphs are strong tools for representing structures formed by pixels, like edges and textures.

The filtering of graph signals is not only an extension of that for standard time- and spatial-domain signals, but it also has its own interesting properties. For example, GSP can represent traditional pixel-dependent image filtering methods as graph spectral domain filters. Furthermore, theory and design methods for wavelets and filter banks, which are studied extensively in signal and image processing, are also updated to treat graph signals.

In this chapter, the spectral-domain filtering of graph signals is introduced. In section 1.2, the filtering of time-domain signals is briefly described as a starting point. The filtering of graph signals, both in the vertex and spectral domains, is detailed in section 1.3, in addition to its relationship with classical filtering. Edge-preserving image smoothing is represented as a graph filter in section 1.4. Furthermore, a framework of filtering by multiple graph filters, i.e. graph wavelets and filter banks, is presented in section 1.5. Eventually, section 1.6 introduces several fast computation methods of graph filtering. Finally, the concluding remarks of this chapter are discussed in section 1.7.

Graph Spectral Image Processing,
coordinated by Gene CHEUNG and Enrico MAGLI.

1.2. Review: filtering of time-domain signals

We start by reviewing the filtering in discrete-time linear time-invariant (LTI) systems, which has been extensively studied in literature. Suppose that a one-dimensional discrete-time signal x_n is obtained by sampling its continuous-time counterpart $x(t)$, with a fixed sampling period T, i.e. $x_n = x(nT)$. A two-dimensional image signal can be similarly obtained by performing sampling in both the horizontal and vertical directions. In this case, the spatial sampling period usually corresponds to the spacing between an array of photosensors.

Suppose that an impulse response of a filter h_n is given *a priori*. The discrete-time filtered signal y_n in the LTI system is calculated from x_n and h_n by convolution as follows:

$$y_n = h_n * x_n := \sum_{k=-\infty}^{\infty} h_{n-k} x_k = \sum_{k=-\infty}^{\infty} x_{n-k} h_k \qquad [1.1]$$

This equation is based on the *shift* of the signal or impulse response. In LTI systems, we (implicitly) assume that the shift of a discrete-time signal is well defined, i.e. x_{n-k} is unique and time invariant. Therefore, equation [1.1] is equivalently represented as

$$\mathbf{y} = \begin{bmatrix} \ddots & & & & & \\ \dots & h_{-1} & h_0 & h_1 & \dots & \\ & \dots & h_{-1} & h_0 & h_1 & \dots \\ & & & & & \ddots \end{bmatrix} \mathbf{x}, \qquad [1.2]$$

where $\mathbf{x} := [\dots, x_{-1}, x_0, x_1, \dots]^\top$ and $\mathbf{y} := [\dots, y_{-1}, y_0, y_1, \dots]^\top$. [1]

In equation [1.2], the impulse response h_k is invariant for n, i.e. the same filter is used for different values of n. Instead, we can use different filters for different values of n to yield y_n, whose impulse response $h_k[n]$ is often defined in a signal-dependent manner, i.e. $h_k[n] \neq h_k[m]$ for $m \neq n$. It is formulated as

$$y_n := \sum_{k=-\infty}^{\infty} h_{n-k}[n] x_k \qquad [1.3]$$

1 Here, we assume both $\mathbf{x}$ and $\mathbf{y}$ are finite length signals and their boundaries are extended or filtered by a boundary filter to ensure that the equation is valid.

and its matrix form representation is

$$\mathbf{y} = \begin{bmatrix} \ddots & & & & & \\ \dots & h_{-1}[n-1] & h_0[n-1] & h_1[n-1] & \dots & \\ & \dots & h_{-1}[n] & h_0[n] & h_1[n] & \dots \\ & & & & & \ddots \end{bmatrix} \mathbf{x}. \quad [1.4]$$

Famous image processing filters in this category include the bilateral filter (Tomasi and Manduchi 1998; Barash 2002; Durand and Dorsey 2002; Fleishman *et al.* 2003), anisotropic diffusion (Weickert 1998; Desbrun *et al.* 1999), adaptive directional wavelets (Chang and Girod 2007; Ding *et al.* 2007; Tanaka *et al.* 2010) and their variants.

It is well known that convolution in the time domain equation [1.1] has an equivalent expression in the frequency (i.e. Fourier) domain as follows:

$$\hat{y}(\omega) = \hat{h}(\omega)\hat{x}(\omega), \quad [1.5]$$

where

$$\hat{x}(\omega) := \sum_{n=-\infty}^{\infty} x_n e^{-j\omega n}. \quad [1.6]$$

Here, $\hat{x}(\omega)$ is the discrete-time Fourier transform (DTFT) of x_n. We utilize the fact that convolution in the time domain is identical to multiplication in the frequency domain. Note that the fixed filter has a corresponding fixed frequency response, and thus, we can intuitively understand the filter characteristics from the frequency response. In contrast, the frequency response of a signal-dependent filter is not always clear in general. Fortunately, this drawback can be partially solved with a graph spectral domain perspective, which is described further.

1.3. Filtering of graph signals

In this chapter, we consider *linear* graph filters. Readers can find nonlinear graph filters, like one used in deep learning, in the following chapters, specifically Chapter 10.

Let us denote a graph filter as $\mathbf{H} \in \mathbb{R}^{N \times N}$, where its elements are typically derived from $\mathcal{G}$ and $\mathbf{x}$. As in the LTI system, the filtered signal is represented as

$$\mathbf{y} = \mathbf{H}\mathbf{x}. \quad [1.7]$$

The representation of its element y_n is similar to that observed in equation [1.3], i.e.

$$y_n := \sum_{k=0}^{N-1} [\mathbf{H}]_{n,k} x_k, \qquad [1.8]$$

where $[\cdot]_{n,k}$ is the n,k-element in the matrix. Similar to discrete-time signals, graph signal filtering may be defined in the vertex and graph frequency domains. These are described in the following.

1.3.1. *Vertex domain filtering*

Vertex domain filtering is an analog of filtering in the time domain. However, GSP systems are not shift-invariant: This means node indices do not have any physical meaning, in general. Therefore, the shift of graph signals based on the indices of nodes, similar to that used for discrete-time signals, would be inappropriate. Moreover, the underlying graph will exhibit a highly irregular connectivity, i.e. the degree in each node will vary significantly. For example, the star graph shown in Figure 1.1 has one center node and $N-1$ surrounding nodes. It is clear that $N-1$ edges are connected to the center node, i.e. the center node has degree $N-1$, whereas all of the surrounding nodes have degree 1. In an image processing perspective, such an irregularity comes from the edge and texture regions. An example is provided on the right side of Figure 1.1. Suppose that we construct a graph based on pixel intensity, i.e. pixels are nodes, and they are connected by edges with higher weights when their pixel values are closer. In this situation, pixels *along* edge/texture directions will be connected to each other strongly with a large degree, whereas those *across* edge/texture directions may have weaker edges, or may even be disconnected. Filtering based on such a graph will therefore reflect structures in the vertex (i.e. pixel) domain.

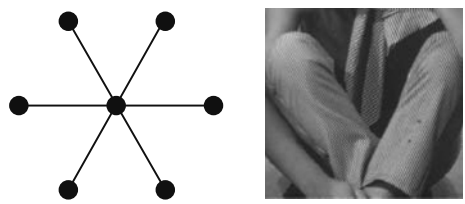

Figure 1.1. *Left: Star graph with $N = 7$. Right: Textured region of Barbara*

Vertex domain filtering can be defined more formally as follows. Let $\mathcal{N}_{n,p}$ be a set of p-hop neighborhood nodes of the nth node. Clearly $|\mathcal{N}_{n,p}|$ varies according to n.

Vertex domain filtering may be typically defined as a local linear combination of the neighborhood samples

$$y_n := \sum_{k \in \mathcal{N}_{n,p}} [\mathbf{H}]_{n,k} x_k. \quad [1.9]$$

Since $\mathcal{N}_{n,p}$ varies according to n, $[\mathbf{H}]_{n,k}$ should be appropriately determined for all n. The matrix form of equation [1.9] may be represented as

$$\mathbf{y} = (\text{diag}([\mathbf{H}]_{0,0}, \ldots, [\mathbf{H}]_{N-1,N-1}) + h(\mathbf{W}))\mathbf{x}, \quad [1.10]$$

where $h(\mathbf{W})$ is a matrix containing filter coefficients $h[n,k]$ $(n \neq k)$ as a function of the adjacency matrix $\mathbf{W}$, in which $[h(\mathbf{W})]_{n,k} = 0$ if $k \notin \mathcal{N}_{n,p}$.

The vertex domain filtering in equations [1.9] and [1.10] requires the determination of $\sum_n |\mathcal{N}_{n,p}|$ filter coefficients, in general; moreover, it sometimes needs increased computational complexity. Typically, $[\mathbf{H}]_{n,k}$ may be parameterized in the following form:

$$\mathbf{y} = \left(\sum_{p=0}^{P-1} h_p \mathbf{W}_p \right) \mathbf{x}, \quad [1.11]$$

where h_p is a real value and $\mathbf{W}_p \in \mathbb{R}^{N \times N}$ is a masked adjacency matrix that only contains p-hop neighborhood elements of $\mathbf{W}$. It is formulated as

$$[\mathbf{W}_p]_{n,k} = \begin{cases} [\mathbf{W}]_{n,k} & \text{if } k \in \mathcal{N}_{n,p}, \\ 0 & \text{otherwise.} \end{cases} \quad [1.12]$$

The number of parameters required in equation [1.12] is P, which is significantly smaller than that required in equation [1.10].

One may find a similarity between the time domain filtering in equation [1.2] and the parameterized vertex domain filtering in equation [1.11]. In fact, if the underlying graph is a cycle graph, equation [1.11] coincides with equation [1.2] with a proper definition of $\mathbf{W}_p$. However, they do not coincide in general cases: It is easily confirmed that the sum of each row of the filter coefficient matrix in equation [1.11] is not constant due to the irregular nature of the graph, whereas $\sum_k h_k$ is a constant in time-domain filtering. Therefore, the parameters of equation [1.11] should be determined carefully.

1.3.2. *Spectral domain filtering*

The vertex domain filtering introduced above intuitively parallels time-domain filtering. However, it has a major drawback in a frequency perspective. As mentioned in section 1.2, time-domain filtering and frequency domain filtering are identical up to the DTFT. Unfortunately, in general, such a simple relationship does not hold in GSP. As a result, the naïve implementation of the vertex domain filtering equation [1.10] does not always have a diagonal response in the graph frequency domain. In other words, the filter coefficient matrix $\mathbf{H}$ is not always diagonalizable by the GFT matrix $\mathbf{U}$, i.e. $\mathbf{U}^\top\mathbf{H}\mathbf{U}$ is not diagonal in general. Therefore, the graph frequency response of $\mathbf{H}$ is not always clear when filtering is performed in the vertex domain. This is a clear difference between the filtering of discrete-time signals and that of the graph signals.

From the above description, we can come up with another possibility for the filtering of graph signals: graph signal filtering defined in the graph frequency domain. This is an analog of filtering in the Fourier domain in equation [1.5]. This spectral domain definition of graph signal filtering has many desirable properties listed as follows:

– diagonal graph frequency response;

– fast computation;

– interpretability of pixel-dependent image filtering as graph spectral filtering.

These properties are described further.

As shown in equation [1.5], the convolution of h_n and x_n in the time domain is a multiplication of $\hat{h}(\omega)$ and $\hat{x}(\omega)$ in the Fourier domain. Filtering in the graph frequency domain utilizes such an analog to define generalized convolution (Shuman *et al.* 2016b):

$$\hat{y}_i := \hat{h}(\lambda_i)\hat{x}_i \qquad [1.13]$$

where $\hat{x}_i = \langle \mathbf{u}_i, \mathbf{x} \rangle$ is the ith GFT coefficient of $\mathbf{x}$ and the GFT basis $\mathbf{u}_i$ is given by the eigendecomposition of the chosen graph operator equation [I.2]. Furthermore, $\hat{h}(\lambda_i)$ is the graph frequency response of the graph filter. The filtered signal in the vertex domain, $y[n]$, can be easily obtained by transforming $\hat{\mathbf{y}}$ back to $y_n = \sum_{i=0}^{N-1} \hat{y}_i [\mathbf{u}_i]_n$, where $[\mathbf{u}_i]_n$ is the nth element of $\mathbf{u}_i$. This is equivalently written in the matrix form as

$$\begin{aligned} \mathbf{y} &= \mathbf{U} \begin{bmatrix} \hat{h}(\lambda_0) & & \\ & \ddots & \\ & & \hat{h}(\lambda_{N-1}) \end{bmatrix} \mathbf{U}^\top \mathbf{x} \qquad [1.14] \\ &= \left(\sum_{i=0}^{N-1} \hat{h}(\lambda_i) \mathbf{P}_{\lambda_i} \right) \mathbf{x}, \end{aligned}$$

where

$$\mathbf{P}_\lambda := \sum_{k \in \sigma(\lambda)} \mathbf{u}_k \mathbf{u}_k^\top \qquad [1.15]$$

is a projection matrix in which $\sigma(\lambda)$ is a set of indices for repeated eigenvalues, i.e. a set of indices such that $\mathbf{L}\mathbf{u}_k = \lambda \mathbf{u}_k$.

For simplicity, let us assume that all eigenvalues are distinct. Under a given GFT basis $\mathbf{U}$, graph frequency domain filtering in equation [1.13] is realized by specifying N graph frequency responses in $\hat{h}(\lambda_i)$. Since this is a diagonal matrix, as shown in equation [1.14], its frequency characteristic becomes considerably clear in contrast to that observed in vertex domain filtering. Note that the naïve realization of equation [1.13] requires specific values of λ_i, i.e. graph frequency values. Therefore, the eigenvalues of the graph operator must be given prior to the filtering. Instead, in this case, we can parameterize a continuous spectral response $\hat{h}(\lambda)$ for the range $\lambda \in [\lambda_{\min}, \lambda_{\max}]$. This graph-independent design procedure has been widely implemented in many spectral graph filters, since the eigenvalues often vary significantly in different graphs.

For the classical Fourier domain filtering, it is enough to consider the frequency range $\omega \in [-\pi, \pi]$ (or an arbitrary 2π interval). However, graph frequency varies according to an underlying graph and/or the chosen graph operator. For example, symmetric normalized graph Laplacians have eigenvalues within $[0, 2]$, whereas combinatorial graph Laplacians do not have such a graph-independent maximum bound. The simple maximum bound of combinatorial graph Laplacian is, for example, given as (Anderson Jr and Morley 1985)

$$\lambda_{N-1} \leq \max\{d_u + d_v | (u, v) \in \mathcal{E}\}, \qquad [1.16]$$

where d_u is the degree of the vertex u. Several other improvements on the bound are also found in literature. Although the graph Laplacians mentioned above have a bound of the largest eigenvalue, such bounds are not applicable to the adjacency matrix. Considering this, appropriate care of the graph frequency range must be taken while designing graph filters.

As mentioned, graph frequency domain filtering is an analog of Fourier domain filtering. However, this does not mean we always obtain a vertex domain expression of this similar to equation [1.9]. Hence, we need to compute the GFT of the input signal, which raises a computational issue described as follows. For the GFT, the eigenvector matrix $\mathbf{U}$ has to be calculated from the graph operator. The eigendecomposition requires $\mathcal{O}(N^3)$ complexity for a dense matrix[2]. This

2 While the computation cost for eigendecomposition of a sparse matrix is generally lower than $\mathcal{O}(N^3)$, it still requires a high computational complexity, especially for large graphs.

calculation often becomes increasingly complex, especially for big data applications, including image processing.

Typically, graph spectral image processing vectorizes image pixels. Let us assume that we have a grayscale image of size $W \times H$ pixels. Its vectorized version is $\mathbf{x} \in \mathbb{R}^{WH}$ and its corresponding graph operator would be $\mathbb{R}^{WH \times WH}$. For example, 4K ultra-high-definition resolution corresponds to $W = 3,840$ and $H = 2,160$, which leads to $WH > 8 \times 10^6$: this is too large to perform eigendecomposition, even for a recent high-spec computer. In section 1.6, several fast computation methods of graph spectral filtering will be discussed to alleviate this problem.

1.3.3. *Relationship between graph spectral filtering and classical filtering*

Filtering in the graph frequency domain seems to be an intuitive extension of Fourier domain filtering into the graph setting. In fact, it coincides with time-domain filtering in a special case, beyond the intuition.

Suppose that the underlying graph is a cycle graph with length N, and its graph Laplacian $\mathbf{L}_{\text{cycle}}$ is assumed as follows:

$$\mathbf{L}_{\text{cycle}} = \begin{bmatrix} 2 & -1 & & & -1 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ -1 & & & -1 & 2 \end{bmatrix}, \qquad [1.17]$$

where its blank elements are zero. It is well known that the eigenvector matrix of $\mathbf{L}_{\text{cycle}}$ is the DFT (Strang 1999), i.e.

$$\mathbf{L}_{\text{cycle}} = \mathbf{U}\mathbf{\Lambda}_{\text{cycle}}\mathbf{U}^* \qquad [1.18]$$

in which

$$\mathbf{u}_k = [1, w^k, w^{2k}, \ldots, w^{(N-1)k}]^\top, \quad w = \exp(2\pi/N). \qquad [1.19]$$

In other words, when we consider a cycle graph and assume its associated graph Laplacian is $\mathbf{L}_{\text{cycle}}$, its GFT is the DFT. Therefore, graph spectral filtering in equation [1.13] is identical to the time-domain filtering. Note that, while $\mathbf{U}$ is the DFT, the interval of its eigenvalues is not equal to $2\pi k/N$. Specificallly, the kth eigenvalue of $\mathbf{L}_{\text{cycle}}$ is $\lambda_k = 2 - 2\cos(2\pi k/N)$.

1.4. Edge-preserving smoothing of images as graph spectral filters

This book (especially this chapter) focuses on graph spectral domain operations for image processing. Here, we describe interconnections between well-studied edge preserving filters and their GSP-based representations. As previously mentioned in this section, pixel-dependent filters do not have frequency domain expressions in a classical sense. This is because the impulse responses vary for different pixel index values n. In the following, we show that such a pixel-dependent filter can be viewed as a graph spectral filter, i.e. it presents a diagonal graph frequency response. Roughly speaking, GSP-based image processing considers the *pixel structure* and the *filter kernel* independently. Therefore, the pixel-dependent processing can be performed with a fixed filter kernel, owing to the underlying graph.

1.4.1. *Early works*

Let us begin with the history before the GSP era. In the mid-1990s, Taubin proposed seminal works on smoothing using graph spectral analysis for 3D mesh processing (Taubin 1995; Taubin *et al.* 1996)[3]. He determined the edge weights of polygon meshes using the Euclidean (geometric) distance between nodes. Assuming $\mathbf{p}_i \in \mathbb{R}^3$ as a 3-D coordinate of the ith node, the edge weight is then defined as

$$W_{i,j} = \eta \cdot \phi(\mathbf{p}_i, \mathbf{p}_j), \qquad [1.20]$$

where η is the normalizing factor and $\phi(\mathbf{p}_i, \mathbf{p}_j)$ is a non-negative function, which assigns a large weight if $\mathbf{p}_i$ and $\mathbf{p}_j$ are close. The typical choice of $\phi(\mathbf{p}_i, \mathbf{p}_j)$ will be $\|\mathbf{p}_i - \mathbf{p}_j\|^{-1}$.

The matrix $\mathbf{W}$ is symmetric. If we choose $\phi(\mathbf{p}_i, \mathbf{p}_i) = 0$, its diagonal elements would become zero, and as a result, $\mathbf{W}$ could be viewed as a normalized adjacency matrix. The coordinates are then smoothed by a graph low-pass filter, after computing the GFT basis $\mathbf{U}$. Similar approaches to this method have been used in several computer graphics/vision tasks (Zhang *et al.* 2010; Vallet and Lévy 2008; Desbrun *et al.* 1999; Fleishman *et al.* 2003; Kim and Rossignac 2005).

For image smoothing, filtering with a heat kernel represented in the graph frequency domain has also been proposed by Zhang and Hancock (2008). In this work, the edge weights of the pixel graph are computed according to photometric distance, i.e. large weights are assigned to the edges whose ends have similar pixel

3 The term "graph signal" was first introduced in Taubin *et al.* (1996), to the best of our knowledge.

values and vice versa. Additionally, the graph spectral filter is defined as a solution for the heat equation on the graph, and is expressed as follows:

$$\hat{h}(\lambda) = e^{-t\lambda}, \qquad [1.21]$$

where $t > 0$ is an arbitrary parameter that helps control the spreading speed caused by diffusion. Note that this method still needs eigendecomposition of the graph Laplacian if we decide to implement equation [1.21] naïvely. Instead, (Zhang and Hancock 2008) represent equation [1.21] using the Taylor series around the origin as follows:

$$e^{-t\lambda} = \sum_{k=0}^{\infty} \frac{t^k}{k!}(-\lambda)^k. \qquad [1.22]$$

By truncating the above equation with an arbitrary order K, we can approximate the heat kernel as a finite-order polynomial (Hammond *et al.* 2011; Shuman *et al.* 2013). In Zhang and Hancock (2008), the Krylov subspace method is used, along with equation [1.22] to approximate the graph filter. The polynomial method for graph spectral smoothing is detailed in section 1.6.5.

Figure 1.2 depicts the approximation error of the heat kernel using the Taylor series. Clearly, its approximation accuracy gets significantly worse when λ is away from 0. Since the maximum eigenvalue $\lambda_{\max}$ highly depends on the graph used, it is better to use different approximation methods like the Chebyshev approximation, which is introduced in section 1.6.

1.4.2. *Edge-preserving smoothing*

Edge-preserving image smoothing is widely used for various tasks, as well as for image restoration (Nagao and Matsuyama 1979; Pomalaza-Raez and McGillem 1984; Weickert 1998; Tomasi and Manduchi 1998; Barash 2002; Durand and Dorsey 2002; Farbman *et al.* 2008; Xu *et al.* 2011; He *et al.* 2013). Image restoration aims to approximate an unknown ground-truth image from its degraded version(s). In contrast, edge-preserving smoothing is typically used to yield a user-desired image from the original one. The resulting image is not necessarily close to the original one.

In the graph setting, we need to define pixel-wise or patch-wise relationships as a distance between pixels or patches, and it is used to construct a graph. The following distances are often considered (Milanfar 2013b), where i and j are pixel or patch indices and $\phi(\cdot)$ is some nonnegative function:

1) Geometric distance: $d_g(i,j) = \phi(\|\mathbf{p}_i - \mathbf{p}_j\|)$, where $\mathbf{p}_i$ is the ith pixel coordinate.

2) Photometric distance: $d_p(i,j) = \phi(\|[\mathbf{x}]_i - [\mathbf{x}]_j\|)$, where $[\mathbf{x}]_i$ is the pixel value (often three dimensional) of the ith pixel/patch.

3) Saliency distance: $d_s(i,j) = \phi(\|s_i - s_j\|)$, where s_i is the ith saliency value.

4) Combinations of the above.

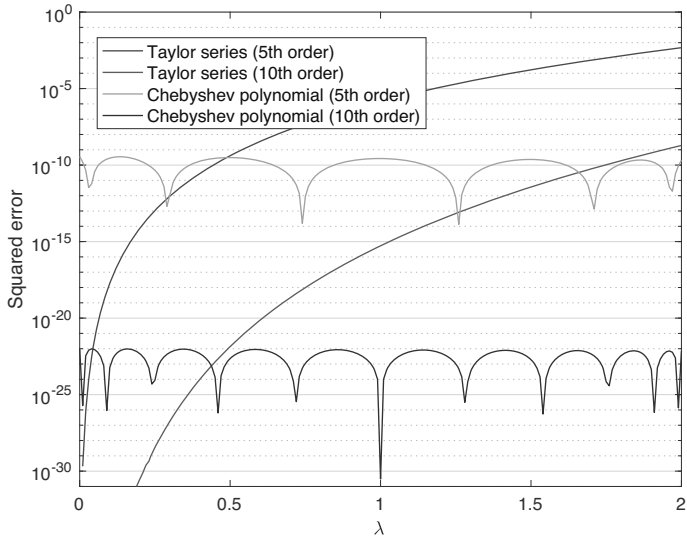

Figure 1.2. *Comparison of approximation errors in $\hat{h}(\lambda) = e^{-\lambda}$. For a color version of this figure, see www.iste.co.uk/cheung/graph.zip*

Saliency of the image/region/pixel is designed to simulate perceptual behavior (Itti *et al.* 1998; Harel *et al.* 2006). A popular choice of $\phi(\cdot)$ is the Gaussian weight

$$\phi(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right), \qquad [1.23]$$

where σ controls the spread of the filter kernel.

Suppose that the filter coefficients are determined based on the above features, and that they are symmetric, i.e. the output pixel value y_i is represented as

$$y_i := \frac{1}{D_i} \sum_j W_{i,j} x_j, \qquad [1.24]$$

where

$$W_{i,j} := \prod_{k=1}^{K} d_k(i,j). \qquad [1.25]$$

Here, $d_k(\cdot,\cdot)$ is one of the distance metrics mentioned earlier and K is the number of features we considered. The scaling factor D_i normalizes the filter weights as $D_i = \sum_j W_{i,j}$. For example, the bilateral filter is $K = 2$ for $d_g(\cdot,\cdot)$ and $d_p(\cdot,\cdot)$.

The Fourier domain representation of such pixel-dependent filters cannot be calculated in a classical sense because it is no longer shift-invariant: the filter matrix $\mathbf{W}$ cannot be diagonalized by the DFT matrix. In contrast, GSP provides a frequency-like notion in the graph frequency domain. In general, the weight matrix $\mathbf{W}$ in equation [1.24] can be regarded as an adjacency matrix because all $d_k(\cdot,\cdot)$ are assumed to be distances between pixels. Suppose that there is no self-loop in $\mathbf{W}$, for simplicity. In general, the smoothed image in equation [1.24] is represented in the following matrix form:

$$\mathbf{y} = \mathbf{D}^{-1}\mathbf{W}\mathbf{x} \qquad [1.26]$$

where $\mathbf{D} = \mathrm{diag}(D_0, \ldots, D_{N-1})$. This can be rewritten by using the relationship $\mathbf{L} = \mathbf{D} - \mathbf{W}$ as (Gadde *et al*. 2013):

$$\mathbf{y} = \mathbf{D}^{-1}(\mathbf{D} - \mathbf{L})\mathbf{x} \qquad [1.27]$$

$$= \mathbf{D}^{-1/2}(\mathbf{I} - \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2})\mathbf{D}^{1/2}\mathbf{x} \qquad [1.28]$$

$$= \mathbf{D}^{-1/2}(\mathbf{I} - \mathbf{L}_n)\underline{\mathbf{x}} \qquad [1.29]$$

where $\underline{\mathbf{x}} := \mathbf{D}^{1/2}\mathbf{x}$ is a degree-normalized signal. Let us denote the eigendecomposition of $\mathbf{L}_n$ as $\mathbf{L}_n := \underline{\mathbf{U}}\underline{\mathbf{\Lambda}}\underline{\mathbf{U}}^\top$. The above filtering in equation [1.29] is further rewritten as:

$$\underline{\mathbf{y}} = \underline{\mathbf{U}}(\mathbf{I} - \underline{\mathbf{\Lambda}})\underline{\mathbf{U}}^\top\underline{\mathbf{x}} \qquad [1.30]$$

$$= \underline{\mathbf{U}}\hat{h}(\underline{\mathbf{\Lambda}})\underline{\mathbf{U}}^\top\underline{\mathbf{x}} \qquad [1.31]$$

$$= h(\mathbf{L}_n)\underline{\mathbf{x}}, \qquad [1.32]$$

where $\underline{\mathbf{y}} := \mathbf{D}^{1/2}\mathbf{y}$ and the graph spectral filter is defined as $\hat{h}(\underline{\lambda}) := 1 - \underline{\lambda}$. Moreover, $\underline{\lambda} \in [0, 2]$ for the symmetric normalized graph Laplacian; therefore, it acts as a linear decay low-pass filter in the graph frequency domain.

This graph spectral representation of a pixel-dependent filter suggests that *the pixel-dependent filter* $\mathbf{W}$ *implicitly and simultaneously designs the underlying graph (and therefore, the GFT basis) and the spectral response of the graph filter*. In other words, the GSP expression of the pixel-dependent filter is free to design the spectral response $\hat{h}(\underline{\lambda})$, apart from the linear decay one, once we determine $\mathbf{W}$. For example, let us consider the following spectral response:

$$\hat{h}(\underline{\lambda}) = \frac{1}{1 + \eta\hat{h}_{\mathrm{HPF}}(\underline{\lambda})}, \qquad [1.33]$$

where $\hat{h}_{\text{HPF}}(\underline{\lambda})$ is an arbitrary graph high-pass filter and $\eta > 0$ is a parameter. In this case, $\hat{h}(\underline{\lambda})$ works as a graph low-pass filter and its spectral shape is controlled by $\hat{h}_{\text{HPF}}(\underline{\lambda})$. In fact, Gadde *et al*. (2013) show that equation [1.33] is the optimal solution for the following signal restoration problem:

$$\arg\min_{\mathbf{x}} \|\mathbf{z} - \mathbf{x}\|_2^2 + \eta \|\mathbf{H}_{\text{HPF}}\mathbf{x}\|_2^2, \qquad [1.34]$$

where $\mathbf{z} = \mathbf{x} + \mathbf{n}$ with additive noise $\mathbf{n}$ and $\mathbf{H}_{\text{HPF}} = \mathbf{U}\hat{h}_{\text{HPF}}(\underline{\mathbf{\Lambda}})\mathbf{U}^\top$.

Image filtering sometimes needs numerous iterations to smooth out the details, in case of textured and/or noisy images. Therefore, to boost up the smoothing effect, the trilateral filter method (Choudhury and Tumblin 2003) first smooths the gradients of the image, and subsequently, the smoothed gradient is utilized to smooth the intensities. Its counterpart in the graph spectral domain is also proposed in Onuki *et al.* (2016) with the parameter optimization method for ρ in equation [1.33], which minimizes MSE after denoising it.

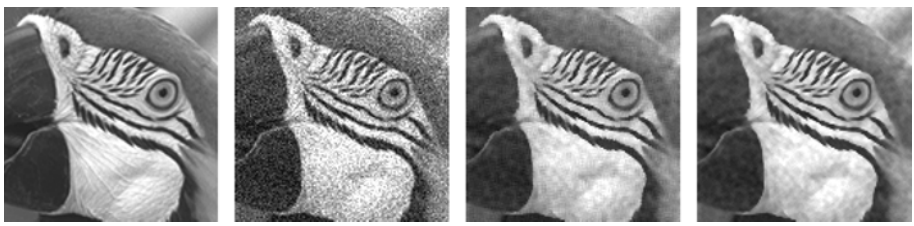

Figure 1.3. *Image denoising example using bilateral filters. From left to right: Original, noisy (PSNR:* 20.02 *dB), bilateral filter in the pixel domain (PSNR:* 26.23 *dB), and bilateral filter in the graph frequency domain (PSNR:* 27.14 *dB). Both bilateral filters use the same parameters*

Figure 1.3 depicts an example of image denoising by the bilateral filter in the graph frequency domain (Gadde *et al*. 2013). The image is degraded by additive white Gaussian noise. The bilateral filter in the graph frequency domain uses the spectral filter parameterized in equation [1.33], with $\hat{h}_{\text{HPF}} = \lambda$ and $\eta = 5$. It is clear that the graph spectral version efficiently removes noise while preserving image edges.

1.5. Multiple graph filters: graph filter banks

In the previous sections, we only considered the case where a single graph spectral filter was applied. Several image processing applications, such as

compression and restoration, often require multiple filters that have different passbands (typically low-pass and high-pass). This signal processing system – so-called filter banks – is also important for GSP. In this section, the spectral domain design of graph filter banks is briefly introduced.

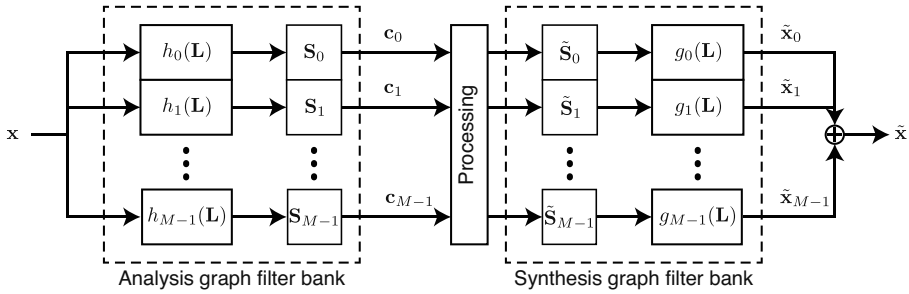

Figure 1.4. *Framework of graph filter bank*

1.5.1. *Framework*

A typical framework of a graph filter bank is illustrated in Figure 1.4. The *analysis transform* decomposes the input signal into some graph frequency components using a set of graph filters $\{h_k(\mathbf{L})\}$ $(k = 0, \ldots, M-1)$. We assume that the graph operator is a graph Laplacian $\mathbf{L}$; however, in general, any graph operator can be applied. The decomposed coefficients (called *transformed coefficients*) are often downsampled by the sampling matrix $\mathbf{S}_k \in \mathbb{R}^{M_k \times N}$, where M_k is the sampling ratio, to reduce the number of coefficients. As a result, the transformed coefficients in each subband are represented as

$$\mathbf{c}_k = \mathbf{S}_k h_k(\mathbf{L})\mathbf{x}. \qquad [1.35]$$

The entire analysis transform is given as follows:

$$\begin{aligned}[\mathbf{c}_0^\top, \ldots, \mathbf{c}_{M-1}^\top]^\top &:= \mathbf{E}\mathbf{x} \qquad [1.36]\\ &= \mathrm{diag}(\mathbf{S}_0, \ldots, \mathbf{S}_{M-1})[h_0^\top(\mathbf{L}), \ldots, h_{M-1}^\top(\mathbf{L})]^\top \mathbf{x}.\end{aligned}$$

The size of $\mathbf{E}$ is $(\sum_k M_k) \times N$ and $\rho := (\sum_k M_k)/N$ is often called the *redundancy* of the transform. The redundancies of transforms are classified as follows:

– $\rho = 1$: *critically sampled* transform. The number of transformed coefficients is the same as N, i.e. the number of elements in $\mathbf{x}$.

– $\rho > 1$: *oversampled* transform. The number of transformed coefficients is larger than N.

– $\rho < 1$: *undersampled* transform. The number of transformed coefficients is smaller than N.

If $\mathbf{S}_k = \mathbf{I}_N$, i.e. no sampling is performed, $\rho = M$, and the transform is called an *undecimated* transform. In general, undersampled transforms will lose the information of the original signal. They cannot recover the original signal $\mathbf{x}$ from the transformed coefficients.

After the analysis transformation, an arbitrary linear and nonlinear operation is performed to $\mathbf{c}_k$ for a target application. For example, small magnitude elements in $\mathbf{c}_k$ are thresholded to denoise or compress the signal. Let us denote $\tilde{\mathbf{c}}_k$ as processed coefficients.

The *synthesis transform* combines $\tilde{\mathbf{c}}_k$ to reconstruct the signal. This is represented as

$$\tilde{\mathbf{x}} := \mathbf{R}[\tilde{\mathbf{c}}_0^\top, \ldots, \tilde{\mathbf{c}}_{M-1}^\top]^\top, \quad [1.37]$$

where $\mathbf{R} \in \mathbb{R}^{N \times (\sum_k M_k)}$ is the synthesis transform matrix. The *perfect reconstruction* transform is defined as the transform that recovers the original signal perfectly, when no processing is performed between the analysis and synthesis transforms. Formally, it satisfies the following condition:

$$\mathbf{REx} = \mathbf{x}. \quad [1.38]$$

The details of perfect reconstruction graph filter banks are provided in the next section.

While $\mathbf{R}$ can be arbitrary, one may need a *symmetric* structure: the synthesis transform represented by multiple filters and upsampling as a counterpart of the analysis transform. In classical signal processing, most filter banks are designed to be symmetric, which, in contrast, is difficult for the graph versions, mainly due to the sampling operations. Several design methods make it possible to design perfect reconstruction graph transforms with a symmetric structure (Narang and Ortega 2012; Narang and Ortega 2013; Shuman *et al.* 2015; Leonardi and Van De Ville 2013; Tanaka and Sakiyama 2014; Sakiyama and Tanaka 2014; Sakiyama *et al.* 2016; Sakiyama *et al.* 2019a; Teke and Vaidyanathan 2016; Sakiyama *et al.* 2019b).

1.5.2. *Perfect reconstruction condition*

Suppose that the redundancy is $\rho \geq 1$ and the columns of $\mathbf{E}$ are linearly independent. The perfect reconstruction condition equation [1.38] is clearly rewritten as

$$\mathbf{RE} = \mathbf{I}_N. \quad [1.39]$$

The critically sampled system constrains that $\mathbf{E}$ is a square matrix; therefore, $\mathbf{R}$ must be $\mathbf{E}^{-1}$ for perfect reconstruction. For the oversampled system, we generally have an infinite number of $\mathbf{R}$ satisfying the condition in equation [1.39]. The most simple and well-known solution is the least squares solution, which is expressed as

$$\mathbf{R} = (\mathbf{E}^\top \mathbf{E})^{-1}\mathbf{E}^\top \qquad [1.40]$$

This is nothing but the Moore–Penrose pseudo inverse of $\mathbf{E}$[4]. This GSP system is generally asymmetric: while the analysis transform has graph filters and possible sampling, the synthesis transform does not have such a clear notion of filtering and upsampling. In general, the asymmetric structure requires a matrix inversion. Additionally, the $N \times N$ matrix $\mathbf{E}^\top \mathbf{E}$ is usually dense, which leads to $\mathcal{O}(N^3)$ complexity.

Therefore, symmetric structures are often desired instead, and they are similar to those that are widely used in classical signal processing. The synthesis transform with a symmetric structure has the following form:

$$\mathbf{R} = [\mathbf{I}_N, \cdots, \mathbf{I}_N]\text{diag}(g_0(\mathbf{L}), \ldots, g_{M-1}(\mathbf{L}))\text{diag}(\tilde{\mathbf{S}}_0, \ldots, \tilde{\mathbf{S}}_{M-1}), \qquad [1.41]$$

where $g_k(\mathbf{L})$ is the kth synthesis filter and $\tilde{\mathbf{S}}_k$ is an upsampling matrix. As a result, each subband has the following input–output relationship:

$$\tilde{\mathbf{x}}_k = g_k(\mathbf{L})\tilde{\mathbf{S}}_k\mathbf{S}_k h_k(\mathbf{L})\mathbf{x}. \qquad [1.42]$$

The resulting output is therefore represented as $\sum_k \tilde{\mathbf{x}}_k$ and for perfect reconstruction, it must be $\mathbf{x}$.

1.5.2.1. *Design of perfect reconstruction transforms: undecimated case*

There are various methods available for designing perfect reconstruction graph transforms. First, let us consider undecimated transforms that exhibit symmetrical structure.

An undecimated transform has no sampling, i.e. $\mathbf{S}_k = \mathbf{I}_N$ for all k. Therefore, the analysis and synthesis transforms, respectively, are represented in the following simple forms:

$$\mathbf{E}_{\text{UD}} = [h_0^\top(\mathbf{L}), \ldots, h_{M-1}^\top(\mathbf{L})]^\top \qquad [1.43]$$

$$\mathbf{R}_{\text{UD}} = [g_0(\mathbf{L}), \ldots, g_{M-1}(\mathbf{L}))]. \qquad [1.44]$$

4 In fact, this $\mathbf{R}$ can also be used for the reconstruction of the undersampled systems.

Accordingly, the perfect reconstruction condition can also be simple. The input–output relationship in equation [1.42] is reduced to

$$\tilde{\mathbf{x}}_k = g_k(\mathbf{L})h_k(\mathbf{L})\mathbf{x}. \quad [1.45]$$

Assuming $p_k(\mathbf{L}) := g_k(\mathbf{L})h_k(\mathbf{L})$ as the kth product filter, the output signal is thus given by

$$\tilde{\mathbf{x}} = \sum_{k=0}^{M-1} p_k(\mathbf{L})\mathbf{x}. \quad [1.46]$$

Therefore, the product filters must satisfy the following condition for perfect reconstruction:

$$\sum_{k=0}^{M-1} p_k(\mathbf{L}) = c\mathbf{I}, \quad [1.47]$$

where c is some constant.

Suppose that $h_k(\mathbf{L})$ and $g_k(\mathbf{L})$ are parameterized as $h_k(\mathbf{L}) = \mathbf{U}\hat{h}_k(\mathbf{\Lambda})\mathbf{U}^\top$ and $g_k(\mathbf{L}) = \mathbf{U}\hat{g}_k(\mathbf{\Lambda})\mathbf{U}^\top$, respectively. In this case, equation [1.47] can be further reduced to

$$\sum_{k=0}^{M-1} \hat{p}_k(\lambda) = c \text{ for all } \lambda \in [\lambda_{\min}, \lambda_{\max}], \quad [1.48]$$

where $\hat{p}_k(\lambda) := \hat{g}_k(\lambda)\hat{h}_k(\lambda)$. This condition is similar to that considered in biorthogonal FIR filter banks in classical signal processing (Vaidyanathan 1993; Vetterli and Kovacevic 1995; Strang and Nguyen 1996). When $\hat{h}_k(\lambda) = \hat{g}_k(\lambda)$ and the filter set satisfies equation [1.48], the filter bank is called a *tight frame* because the perfect reconstruction condition can be rewritten as

$$\sum_{k=0}^{M-1} |\hat{h}_k(\lambda)|^2 = c. \quad [1.49]$$

If $c = 1$, the frame is called a *Parseval frame*. In this case, it conserves the energy of the original signal in the transformed domain. Tight spectral graph filter banks can be constructed by employing the design methods of tight frames in classical signal processing. Examples can be found in Leonardi and Van De Ville (2013); Shuman *et al.* (2015); Sakiyama *et al.* (2016).

1.5.2.2. *Design of perfect reconstruction transforms: decimated case*

Constructing perfect reconstruction graph transforms with sampling is much more difficult than the undecimated version. However, it is required as the storage cost can be increased tremendously for the undecimated versions, especially for signals on a very large graph. Though the general condition is given by equation [1.42], the challenges are designing and choosing the appropriate sampling operator $\mathbf{S}_k$ and the appropriate filters $h_k(\mathbf{L})$ and $g_k(\mathbf{L})$. The perfect reconstruction condition can be satisfied with proper sets of these.

Various methodologies have been proposed in literature with different strategies. The recent methods of such transforms are summarized in Sakiyama *et al.* (2019c). We have omitted these details because they are beyond the scope of this chapter. Instead, some design guidelines are listed as follows:

– Sampling operator: In GSP, two different definitions of sampling operators exist (Tanaka *et al.* 2020; Tanaka 2018; Tanaka and Eldar 2020). One is vertex domain sampling, which is an analog of time-domain sampling. The other is graph frequency domain sampling, which is a counterpart of the Fourier domain expression of sampling. Both are not interchangeable in general, and have their own advantages and disadvantages.

– Localized transform: As mentioned later in section 1.6.5, polynomial filters are localized in the vertex domain. Furthermore, if all filters are polynomials, the entire transform can be eigendecomposition free, and thus, its computation decreases significantly.

– Flexible design adapting various spectra: Different graphs have unique eigenvalue distributions, and different graph operators have unique characteristics. Additionally, we sometimes encounter repeated eigenvalues. A unified design strategy is required for representing various graph signals sparsely.

1.6. Fast computation

As we mentioned in the previous sections, a naïve implementation of spectral graph filters often requires a high computational complexity. Since we often need to process high-resolution images, reducing such a complexity would be a high priority. Moreover, spectral filtering is often iteratively employed for image restoration, like an internal algorithm for convex optimization problems. In this section, we describe a workaround to alleviate computational burden for graph spectral filtering.

1.6.1. *Subdivision*

Digital image processing has a long history, and the subdivision of images has been widely used for various image processing tasks. For example, JPEG and MPEG

image/video compression standards still use block-based predictions and transforms, even in their most recent standards. Moreover, graph-based image processing also uses such a subdivision as preprocessing. It can also be combined with the following fast computation approaches.

A simple solution for image subdivision is the block-based approach. It divides the input image into an equal-sized subblocks (these blocks can be overlapped with an appropriate window function), after which the favorite image processing tasks can be performed. The advantage it provides is simplicity: we only have to consider the size of subblocks to make a trade-off between performance and complexity. Sizes of the consistent image regions vary significantly; as a result, a recursive subdivision, called quadtree decomposition, provides a good trade-off.

More complex image subdivisions are also possible by utilizing an image segmentation. Although these segmentated sub-images are not rectangular in general, we can directly perform graph-based image processing in such non-rectangular regions by using appropriate graphs.

1.6.2. *Downsampling*

Downsampling is another typical approach for reducing the computation cost, and this has been used everywhere in image processing applications. A challenge for graph-based image processing is to find a good low-resolution graph, which properly reflects the original pixel structure.

Reducing the size of a graph is called *graph reduction* or *graph coarsening*. It is divided into two phases:

1) Phase 1: reducing the number of nodes;

2) Phase 2: reconnecting edges for the downsampled pixels.

In image processing, Phase 1 is relatively straightforward. We can assume the original image pixels are nodes on a uniform grid. As in usual image processing, picking up every other node (when the image signal is downsampled by two) will be reasonable.

For more general graphs like those used in point cloud processing, we need to select the "best" set of nodes. This problem is called *sampling set selection*. Although this is beyond our scope in this chapter, please refer to (Tanaka *et al*. 2020; Sakiyama *et al*. 2019c) and references therein.

In contrast to Phase 1, Phase 2 is not very straightforward. If pixels in the original image/block are associated with a graph, the downsampled one should be reconnected by edges, such that the reduced-size graph reflects the original structure. In the general

GSP study, desiderata for the reduced-size graphs have been suggested in Shuman *et al.* (2016a) as follows:

1) the reduced-size graph has non-negative edge weights;

2) the connectivity of the original graph is preserved in the reduced-size graphs;

3) the spectrum of the reduced-size graph is representative of the original graph;

4) the reduced-size graph preserves the original structural properties;

5) if two nodes are connected in the original graph, they should have a similar edge weights in the reduced-size graph;

6) it is tractable in terms of implementation and computational complexity;

7) the reduced-size graph preserves the sparsity (i.e. the ratio between the number of nonzero edges and that of pixels) of the original one.

Existing reconnection methods do not always satisfy all of these simultaneously; however, they do exhibit some of these properties. The order of the desired properties depends on applications considered. Major approaches have been summarized in Shuman *et al.* (2016a).

1.6.3. *Precomputing GFT*

If an image or subblock has several typical patterns, i.e. graphs, precomputing GFT bases for these graphs may be a reasonable choice to decrease the computational burden. This is because when we use them off the shelf, the computation cost reduces significantly as a result of decreased $\mathcal{O}(N^3)$ complexity, with a sacrifice of the storage cost for the GFT matrices and the cost of searching the optimal precomputed GFT from a given image. This precomputing strategy is popular in standard image processing. For example, in modern image/video coding standards, some precomputed transforms, such as DCT and discrete sine transform (DST) with various sizes, are utilized to represent image blocks as sparsely as possible.

Some precomputing methods have been proposed by Hu *et al.* (2015) and Zhang and Liang (2017), and they are mainly used for image compression. As expected, the GFT yields sparse transformed coefficients for piecewise smooth images/blocks. For those without such piecewise regions, conventional transforms like the DCT and DST are basically included as a set of precomputed bases.

1.6.4. *Partial eigendecomposition*

To emphasize, the eigendecomposition of the graph operator will need $\mathcal{O}(N^3)$ complexity in general. In other words, we can reduce the complexity if we can

assume graph signals on the underlying graph are bandlimited. Suppose that the signal is K-bandlimited, which is typically defined as

$$\|\hat{\mathbf{x}}\|_0 \leq K, \tag{1.50}$$

where $\|\cdot\|_0$ represents the number of non-zero elements, i.e. ℓ_0 pseudo-norm. Here, without loss of generality, we can assume the first K GFT coefficients are non-zero:

$$\hat{x}_i = 0 \text{ for } i > K. \tag{1.51}$$

With the GFT basis $\mathbf{U}$, it is equivalently represented as

$$\hat{\mathbf{x}} = \mathbf{U}^\top \mathbf{x} = [\underbrace{\times \quad \dots \quad \times}_{K} \underbrace{0 \quad \dots \quad 0}_{N-K}]^\top = [(\mathbf{U}_K^\top \mathbf{x})^\top, \mathbf{0}^\top]^\top, \tag{1.52}$$

where $\times$ represents some possible non-zero elements and

$$\mathbf{U}_K := \left[\mathbf{u}_0 \ \dots \ \mathbf{u}_{K-1}\right]. \tag{1.53}$$

A partial eigendecomposition proposed in literature gives the following approximation of $\mathbf{L}$:

$$\tilde{\mathbf{L}} := \mathbf{U}_K \text{diag}(\lambda_0, \dots, \lambda_{K-1}) \mathbf{U}_K^\top. \tag{1.54}$$

Evaluating $h(\tilde{\mathbf{L}})\mathbf{x}$ only requires K ($< N$) eigenvectors and eigenvalues, which is significantly less than that obtained using the full eigendecomposition. In general, its computational complexity will be $\mathcal{O}(KN^2)$.

1.6.5. *Polynomial approximation*

The previous subsection proposes that we can alleviate the heavy computational burden by assuming the bandlimitedness of the graph signal. However, this requires the assumption on the signal model prior to filtering, but the signal is *not* bandlimited in general.

In many application scenarios, we often only need the *evaluation* of $\mathbf{x}$ with a given (linear) matrix function $h(\mathbf{L})$. That is, the *eigenvalues and eigenvectors themselves are often unnecessary*. The polynomial approximation methods introduced here enable us to calculate an approximation of $\mathbf{y} = h(\mathbf{L})\mathbf{x}$ without the (partial) decomposition of the variation operator.

Another advantage of filtering using a polynomial filter function is the vertex localization. The local filtering could capture local variations of pixel values, which

are generally preferable. In contrast, filtering in the graph frequency domain (equation [1.13]) is usually not localized in the vertex domain, because eigenvectors often have global support on the graph. Therefore, localizing graph filter response, both in the vertex and graph frequency domains, has been studied extensively (Shuman *et al*. 2013; Shuman *et al.* 2016b; Sakiyama *et al.* 2016). In fact, the localization of graph spectral filters can be controlled using polynomial filtering.

Polynomial graph filters are defined as follows:

$$h(\mathbf{L}) := \sum_{k=0}^{K} c_k \mathbf{L}^k, \tag{1.55}$$

where c_k is the kth order coefficient of the polynomial. It is known that each row of $\mathbf{L}^k$ collects its k-hop neighborhood; therefore, equation [1.55] is exactly the K-hop localized in the vertex domain. Note that $\mathbf{L}^k$ can be represented as

$$\mathbf{L}^k = (\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top)^k = \mathbf{U}\mathbf{\Lambda}^k\mathbf{U}^\top = \mathbf{U}\begin{bmatrix} \lambda_0^k & & \\ & \ddots & \\ & & \lambda_{N-1}^k \end{bmatrix}\mathbf{U}^\top. \tag{1.56}$$

Here, we utilized the orthogonality of $\mathbf{U}$. We can rewrite equation [1.55] by using equation [1.56] as:

$$h(\mathbf{L}) = \sum_{k=0}^{K} c_k \mathbf{L}^k = \mathbf{U}\begin{bmatrix} \sum_k c_k\lambda_0^k & & \\ & \ddots & \\ & & \sum_k c_k\lambda_{N-1}^k \end{bmatrix}\mathbf{U}^\top. \tag{1.57}$$

Consequently, the polynomial graph filter has the following graph frequency response:

$$\hat{h}(\lambda) = c_0 + c_1\lambda + c_2\lambda^2 + \cdots + c_K\lambda^K = \sum_{k=0}^{K} c_k\lambda^k. \tag{1.58}$$

Especially, the output signal in the vertex domain is given by

$$\mathbf{y} = h(\mathbf{L})\mathbf{x} = \left(\sum_{k=0}^{K} c_k\mathbf{L}^k\right)\mathbf{x}. \tag{1.59}$$

This indicates that we do not need to compute specific eigenvalues and eigenvectors for just calculating $\mathbf{y}$. Specifically, we need to evaluate $\mathbf{L}\mathbf{x}, \mathbf{L}^2\mathbf{x}, \ldots, \mathbf{L}^K\mathbf{x}$. Calculating $\mathbf{L}\mathbf{z}$, where $\mathbf{z}$ is an arbitrary vector, requires $\mathcal{O}(|\mathcal{E}|)$

complexity. Additionally, $\mathcal{O}(N)$ is required for computing $c_k\mathbf{L}^k\mathbf{x}$ (and it is repeated K times). As a result, the entire complexity will be $\mathcal{O}(K(|\mathcal{E}| + N))$. It is usually much lower than the partial eigendecomposition. In general, $K \ll |\mathcal{E}|$; therefore, the number of edges is a dominant factor affecting the complexity.

Suppose that a fast computation is required for the spectral response of a graph filter $\hat{h}(\lambda)$, which is not a polynomial. Based on equation [1.59], we can approximate the output $\mathbf{y}$ if $\hat{h}(\lambda)$ is satisfactorily approximated by a polynomial.

Any polynomial approximation methods, e.g. Taylor expansion, are possible for the above-mentioned polynomial filtering. In GSP, Chebyshev polynomial approximation is implemented frequently. The Chebyshev expansion gives an approximate minimax polynomial, i.e. the maximum approximation error can be reduced.

The approximated version of $h(\mathbf{L})\mathbf{x}$ by the Kth order shifted Chebyshev polynomial, $h_{\text{Cheb}}(\mathbf{L})\mathbf{x}$, is given by Shuman *et al.* (2013); Hammond *et al.* (2011)

$$h_{\text{Cheb}}(\mathbf{L})\mathbf{x} = \left\{ \frac{1}{2}c_0 + \sum_{k=1}^{K} c_i \bar{T}_k(\mathbf{L}) \right\} \mathbf{x} \qquad [1.60]$$

and it has the recurrence property:

$$\bar{T}_k(\mathbf{L}) = \frac{4}{\lambda_{\max}}(\mathbf{L} - \mathbf{I})\bar{T}_{k-1}(\mathbf{L}) - \bar{T}_{k-2}(\mathbf{L}) \qquad [1.61]$$

with $\bar{T}_0(\mathbf{L}) = \mathbf{I}$ and $\bar{T}_1(\mathbf{L}) = 2(\mathbf{L} - \mathbf{I})/\lambda_{\text{max}}$. The kth Chebyshev coefficient c_k is defined as

$$c_k = \frac{2}{P} \sum_{p=1}^{P} \cos\left(\frac{k\left(p - \frac{1}{2}\right)\pi}{P} \right) \hat{h}\left(\frac{\lambda_{\max}}{2} \left(\cos\left(\frac{\left(p - \frac{1}{2}\right)\pi}{P} \right) + 1 \right) \right) \qquad [1.62]$$

for $k = 0, \ldots, K$, where P is the number of sampling points used to compute the Chebyshev coefficients and is usually set to $P = K + 1$. The approximated filter in equation [1.60] is clearly a Kth order polynomial of λ. As a result, it is K-hop localized in the vertex domain, as previously mentioned (Shuman *et al.* 2011; Hammond *et al.* 2011).

The approximation error for the Chebyshev polynomial has been well studied in the context of numerical computation (Vetterli *et al.* 2014; Phillips 2003):

THEOREM 1.1.– Let K be the polynomial degree of the Chebyshev polynomial and assume that $\hat{h}(\xi)$ has $(K+1)$ continuous derivatives on $[-1, 1]$. In this case, the upper bound of the error is given as follows:

$$|E_{\max,K}| \leq \frac{1}{2^K(K+1)!} \max \left| \frac{d^{(K+1)}}{d\xi^{(K+1)}} \hat{h}\left(\xi\right) \right|. \qquad [1.63]$$

1.6.6. *Krylov subspace method*

The Krylov subspace $\mathcal{K}_K$ of an arbitrary square matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ and a vector $\mathbf{x}$ is defined as follows:

$$\mathcal{K}_K := \text{span}\{\mathbf{x}, \mathbf{W}\mathbf{x}, \mathbf{W}^2\mathbf{x}, \ldots, \mathbf{W}^{K-1}\mathbf{x}\} \quad [1.64]$$

The Krylov subspace method, in terms of GSP, refers to filtering, i.e. evaluating an arbitrary filtered response $h(\mathbf{W})\mathbf{x}$, realized in a Krylov subspace $K \ll N$. Many methods to evaluate $h(\mathbf{W})\mathbf{x}$ in a Krylov subspace have been proposed, mainly in computational linear algebra and numerical computation (Golub and Van Loan 1996). A famous approximation method is the Arnoldi approximation, which is given by

$$\mathbf{y} \sim \beta \mathbf{V}_K h(\mathbf{H}_K)[1, 0, \ldots, 0]^\top, \quad [1.65]$$

where $h(\mathbf{H}_K)$ is evaluating $h(\cdot)$ for the upper Heisenberg matrix $\mathbf{H}_K$, which is obtained by using the Arnoldi process. Furthermore, $\mathbf{H}_K$ is expected to be much smaller than the original matrix; therefore, evaluating $h(\mathbf{H}_K)$ using full eigendecomposition will be feasible and light-weighted.

1.7. Conclusion

This chapter introduces the filtering of graph signals performed in the graph frequency domain. This is a key ingredient of graph spectral image processing presented in the following chapters. The design methods of efficient and fast graph filters and filter banks, along with *fast GFT* (such attempts can be found in Girault *et al*. (2018); Lu and Ortega (2019)), are still a vibrant area of GSP: the chosen graph filters directly affect the quality of processed images. This chapter only provided a brief overview of graph spectral filtering. Please refer to the references for more details.

1.8. References

Anderson Jr. W.N. and Morley, T.D. (1985). Eigenvalues of the Laplacian of a graph. *Linear and Multilinear Algebra*, 18(2), 141–145.

Barash, D. (2002). Fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(6), 844–847.

Chang, C.-L. and Girod, B. (2007). Direction-adaptive discrete wavelet transform for image compression. *IEEE Trans. Image Process.*, 16(5), 1289–1302.

Choudhury, P. and Tumblin, J. (2003). The trilateral filter for high contrast images and meshes. *Eurographics Rendering Symposium*, 186–196.

Desbrun, M., Meyer, M., Schröder, P., Barr, A.H. (1999). Implicit fairing of irregular meshes using diffusion and curvature flow. *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, 317–324.

Ding, W., Wu, F., Wu, X., Li, S., Li, H. (2007). Adaptive directional lifting-based wavelet transform for image coding. *IEEE Trans. Image Process*, 16(2), 416–427.

Durand, F. and Dorsey, J. (2002). Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics (TOG)*, 21, 257–266.

Farbman, Z., Fattal, R., Lischinski, D., Szeliski, R. (2008). Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Transactions on Graphics (TOG)*, 27, 67.

Fleishman, S., Drori, I., Cohen-Or, D. (2003). Bilateral mesh denoising. *ACM Transactions on Graphics (TOG)*, 22, 950–953.

Gadde, A., Narang, S.K., Ortega, A. (2013). Bilateral filter: Graph spectral interpretation and extensions. *IEEE International Conference on Image Processing*, 1222–1226.

Girault, B., Ortega, A., Narayanan, S. (2018). Irregularity-aware graph Fourier transforms. *IEEE Transactions on Signal Processing*, 66(21), 5746–5761.

Golub, G.H. and Van Loan, C.F. (1996). *Matrix Computations*, Johns Hopkins University Press, Maryland.

Hammond, D.K., Vandergheynst, P., Gribonval, R. (2011). Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2), 129–150.

Harel, J., Koch, C., Perona, P. (2006). Graph-based visual saliency. *Proceedings of the 19th International Conference on Neural Information Processing Systems*, 545–552.

He, K., Sun, J., Tang, X. (2013). Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6), 1397–1409.

Hu, W., Cheung, G., Ortega, A., Au, O.C. (2015). Multiresolution graph Fourier transform for compression of piecewise smooth images. *IEEE Transactions on Image Processing*, 24(1), 419–433.

Itti, L., Koch, C., Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(11), 1254–1259.

Kim, B.-M. and Rossignac, J. (2005). Geofilter: Geometric selection of mesh filter parameters. *Computer Graphics Forum*, 24, 295–302.

Leonardi, N. and Van De Ville, D. (2013). Tight wavelet frames on multislice graphs. *IEEE Trans. Signal Process.*, 16(13), 3357–3367.

Lu, K.-S. and Ortega, A. (2019). Fast graph Fourier transforms based on graph symmetry and bipartition. *IEEE Trans. Signal Process.*, 67(18), 4855–4869.

Milanfar, P. (2013b). A tour of modern image filtering. *IEEE Signal Processing Magazine*, 30(1), 106–128.

Nagao, M. and Matsuyama, T. (1979). Edge preserving smoothing. *Computer Graphics and Image Processing*, 9(4), 394–407.

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

Narang, S.K. and Ortega, A. (2013). Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs. *IEEE Transactions on Signal Processing*, 61(19), 4673–4685.

Onuki, M., Ono, S., Yamagishi, M., Tanaka, Y. (2016). Graph signal denoising via trilateral filter on graph spectral domain. *IEEE Trans. Signal Inf. Process. Netw.*, 2(2), 137–148.

Phillips, G.M. (2003). *Interpolation and Approximation by Polynomials*. Springer, New York.

Pomalaza-Raez, C. and McGillem, C. (1984). An adaptative, nonlinear edge-preserving filter. *IEEE Trans. Acoust., Speech, Signal Process*, 32(3), 571–576.

Sakiyama, A. and Tanaka, Y. (2014). Oversampled graph Laplacian matrix for graph filter banks. *IEEE Trans. Signal Process*, 62(24), 6425–6437.

Sakiyama, A., Watanabe, K., Tanaka, Y. (2016). Spectral graph wavelets and filter banks with low approximation error. *IEEE Trans. Signal Inf. Process. Netw.*, 2(3), 230–245.

Sakiyama, A., Tanaka, Y., Tanaka, T., Ortega, A. (2019a). Eigendecomposition-free sampling set selection for graph signals. *IEEE Trans. Signal Process*, 67(10), 2679–2692.

Sakiyama, A., Watanabe, K., Tanaka, Y. (2019b). m-channel critically sampled spectral graph filter banks with symmetric structure. *IEEE Signal Processing Letters*, 26(5), 665–669.

Sakiyama, A., Watanabe, K., Tanaka, Y., Ortega, A. (2019c). Two-channel critically-sampled graph filter banks with spectral domain sampling. *IEEE Trans. Signal Process*, 67(6), 1447–1460.

Shuman, D.I., Vandergheynst, P., Frossard, P. (2011). Chebyshev polynomial approximation for distributed signal processing. *Proc. DCOSS'11*, 1–8.

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

Shuman, D.I., Wiesmeyr, C., Holighaus, N., Vandergheynst, P. (2015). Spectrum-adapted tight graph wavelet and vertex-frequency frames. *IEEE Trans. Signal Process*, 63(16), 4223–4235.

Shuman, D.I., Faraji, M.J., Vandergheynst, P. (2016a). A multiscale pyramid transform for graph signals. *IEEE Trans. Signal Process*, 64(8), 2119–2134.

Shuman, D.I., Ricaud, B., Vandergheynst, P. (2016b). Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis*, 40(2), 260–291.

Strang, G. (1999). The discrete cosine transform. *SIAM Rev.*, 41(1), 135–147.

Strang, G. and Nguyen, T.Q. (1996). *Wavelets and Filter Banks*. Wellesley-Cambridge, Massachusetts.

Tanaka, Y. (2018). Spectral domain sampling of graph signals. *IEEE Trans. Signal Process*, 66(14), 3752–3767.

Tanaka, Y. and Eldar, Y.C. (2020). Generalized sampling on graphs with subspace and smoothness priors. *IEEE Transactions on Signal Processing*, 68, 2272–2286.

Tanaka, Y. and Sakiyama, A. (2014). M-channel oversampled graph filter banks. *IEEE Trans. Image Process*, 62(14), 3578–3590.

Tanaka, Y., Hasegawa, M., Kato, S., Ikehara, M., Nguyen, T.Q. (2010). Adaptive directional wavelet transform based on directional prefiltering. *IEEE Trans. Image Process.*, 19(4), 934–945.

Tanaka, Y., Eldar, Y.C., Ortega, A., Cheung, G. (2020). Sampling signals on graphs: From theory to applications. *IEEE Signal Processing Magazine*, 37(6), 14–30.

Taubin, G. (1995). A signal processing approach to fair surface design. *Proc. SIGGRAPH'95*, 351–358.

Taubin, G., Zhang, T., Golub, G.H. (1996). Optimal surface smoothing as filter design. *Proc. ECCV'96*, 283–292.

Teke, O. and Vaidyanathan, P.P. (2016). Extending classical multirate signal processing theory to graphs – Part II: M-channel filter banks. *IEEE Trans. Image Process*, 65(2), 423–437.

Tomasi, C. and Manduchi, R. (1998), Bilateral filtering for gray and color images. *IEEE International Conference on Computer Vision*, 839–846.

Vaidyanathan, P.P. (1993). *Multirate Systems and Filter Banks*. Prentice Hall, New Jersey.

Vallet, B. and Lévy, B. (2008). Spectral geometry processing with manifold harmonics. *Computer Graphics Forum*, 27, 251–260.

Vetterli, M. and Kovacevic, J. (1995). *Wavelets and Subband Coding*. Prentice Hall, New Jersey.

Vetterli, M., Kovacevi, J., Goyal, V.K. (2014). *Foundations of Signal Processing*. Cambridge University Press, Cambridge.

Weickert, J. (1998). *Anisotropic Diffusion in Image Processing 1*, Teubner, Stuttgart.

Xu, L., Lu, C., Xu, Y., Jia, J. (2011). Image smoothing via $l0$ gradient minimization. *ACM Transactions on Graphics (TOG)*, 30, 174.

Zhang, F. and Hancock, E.R. (2008). Graph spectral image smoothing using the heat kernel. *Pattern Recognition*, 41(11), 3328–3342.

Zhang, D. and Liang, J. (2017). Graph-based transform for 2D piecewise smooth signals with random discontinuity locations. *IEEE Transactions on Image Processing*, 26(4), 1679–1693.

Zhang, H., Van Kaick, O., Dyer, R. (2010). Spectral mesh processing. *Computer Graphics Forum*, 29, 1865–1894.