Master Thesis

A Laplacian Pyramid Scheme in Graph Signal Processing

Mohammad Javad Faraji

Signal Processing Laboratory (LTS2)
École Polytechnique Fédérale de Lausanne

Supervisor
Prof. Pierre Vandergheynst

June 2011

Abstract

In this report we extend the ideas behind classical multiscale signal processing techniques in order to analyze data residing on graphs. In particular, we extend the notions of filtering, downsampling, and upsampling to functions defined on graphs. We then use these notions to define a Laplacian pyramid scheme that generates a multiscale transform for signals on graphs. Possible applications of our proposed transform include coding, denoising, and function recovery which are among the most important tasks in signal processing.

Acknowledgments

I would like to express the deepest appreciation to my advisor, Prof. Pierre Vandergheynst. It has been a great pleasure for me to know you and work with you during my several projects in your lab. Thank you for accepting me in LTS2 and for the enthusiasm and inspiration you have been always giving me.

I would also like to express profound gratitude to David I. Shuman for his continuous support during the project. Thank you for your helpful guidance and hours you spent for me.

I am as ever, especially indebted to my family and my lovely fiancée. Thank you for your unconditional love, support and understanding throughout my studies. I would like to dedicate this thesis to you.

Contents

List of Figures

Chapter 1

Introduction

Although many interesting problems in signal processing involve analyzing structured data such as time series data and images that consist of sampled real-valued functions defined on regular Euclidean spaces, many other interesting applications involve data defined on more topologically complicated domains such as *network-like structures*, manifolds, or irregularly shaped domains. Sensor networks, computer networks, and transportation networks are just a few applications where one may find such data.

In order to mathematically analyze data defined on network-like structures, we can use graph theory to model data. As an example, consider sensor networks which are now prevalent in many applications, such as environmental monitoring and medical diagnostics. The sensor nodes are often deployed to collectively achieve tasks such as estimation, detection, and classification. To analyze data captured by individual sensors, we first need to model the network by a graph where each vertex corresponds to an individual sensor and then consider a signal containing the information content of sensors as a *graph-signal* defined as a vector on the vertices of graph.

Many signal processing techniques are based on transform methods in which the signals are represented in another basis before analyzing. One of the most important signal processing techniques in use is *multiscale analysis*. In multiscale analysis, data is captured in hierarchical structures where each level corresponds to a reduced-resolution approximation. The Laplacian Pyramid (LP) proposed by Burt and Adelson [1] is one of the earliest examples of such a scheme. It is used especially in image coding. Wavelet analysis, another popular mathematical tool in signal processing, is also based on multiscale analysis. Because of the power of wavelet analysis in sparse approximation of signals, it is very popular in many signal processing appli-

cations such as coding, denoising, and function recovery. However, graphs do not in general have the regularity properties of the Euclidean spaces where many traditional sampled signals in time and space lie. Therefore, it is not straightforward how to extend transforms designed for these traditional signals to transforms for graph signals.

Since the importance of analyzing network-like structured data is increasing, many researchers have recently focused on developing new techniques for graph signal processing. In the graph literature, a significant amount of work has been done to define transforms and filter-banks on graphs [2, 3]. In this report, we are going to focus on multiscale analysis of graph-signals. How we could develop the ideas in classical multiresolution signal processing for the space of the functions defined on graphs.

The remainder of the report is as follows. In the next chapter, we talk about basic concepts of weighted graphs and provide the notations necessary to model network-like structured data. In Chapter 3, we review the classical Laplacian pyramid framework. Then after proposing some preliminary tools in Chapter 4 for manipulating data on graphs, we extend the classical LP to a Laplacian pyramid scheme for signals on graphs in chapter 5. We also discuss two important implementation issues in the graph LP structure. Chapter 6 extend the single-layer graph LP scheme to its multi-layer version. In this chapter, we also talk about Kron reduction, a graph reduction technique used in graph downsampling. Finally, the conclusions and future works are presented.

Chapter 2

Weighted Graphs

In order to analyze network-like structured data, nothing is more important than modeling data by means of a graph. So, first we are going to review the concepts of weighted graphs that can be used for describing networks. In the following we focus on notations and the characteristics of weighted graphs in spectral graph theory.

2.1 Preliminaries and Notations

A weighted graph $G = \{E, V, \omega\}$ is a set of vertices V, edges E, and a weighting function $\omega : E \to \mathcal{R}^+$, that assigns a positive real number to each edge if there exist such an edge. This number is somehow a characterization of the information between corresponding vertices, i.e. those vertices which are connected by the edge. The adjacency matrix $A = [a_{ij}]_{n\times n}$ of a weighted graph is an $n \times n$ matrix where n is the number of vertices, i.e. $|V| = n < \infty$ and the entries of this matrix are the weights of edges where these edges exist, and zeros elsewhere. In other words, $a_{ij} = \omega_{ij}\delta_{i\leftrightarrow j}$ where ω_{ij} is the weight between vertices i and j and $\delta_{i\leftrightarrow j}$ is the indicator function which is equal to one for connected i, j and zero otherwise.

The Laplacian operator is then defined by

$$\mathcal{L} = D - A, \tag{2.1}$$

where $D = diag\{\{d_i\}_{i=1}^n\}$ is an $n \times n$ diagonal matrix with on-diagonal entries of $d_i = \sum_j a_{ij}$ which indicate the degree of each vertex, defined as the sum of the weights of all edges incident to it.

Now, consider the space of the functions $f : V \to \mathcal{R}^n$ defined on the vertices of a graph. A graph-signal f can be defined as a set of scalars, where each scalar is assigned to one of the vertices of the graph. In fact, each function can be viewed as an n-tuple vector in an Euclidean space $\mathcal{R}^n$. It can be easily shown that if the Laplacian operator acts on a function f at vertex j, it actually computes the weighted summation of all differences in the value of the function at neighboring vertices with respect to the vertex j. By neighboring, we mean those vertices which are connected to the vertex j. As it is appears in (2.2) the summation over $j \leftrightarrow i$ indicates summation over all vertices i that are connected to the vertex j.

$$\mathcal{L}f(j) = \sum_{j \leftrightarrow i} \omega_{ji}(f(j) - f(i)). \tag{2.2}$$

2.2 Spectral Characteristics

In graph theory, the spectral domain is spanned by the eigenvectors of the Laplacian operator $\mathcal{L}$. We denote the eigenvalues of $\mathcal{L}$ with $\{\lambda_l\}_{l=0}^{n-1}$. As $\mathcal{L}$ is symmetric, each of eigenvalues λ_l is real. For the graph Laplacian, it can be shown that the eigenvalues are all non-negative and 0 appears as an eigenvalue with multiplicity equal to the number of connected subgraphs [4]. Hence, for a connected graph, we can order the eigenvalues such that

$$0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_{n-1}. \tag{2.3}$$

We also denote the eigenvector corresponding to λ_l with $\mathbf{v}_l$ where $\mathbf{v}_l$ satisfies

$$\mathcal{L}\mathbf{v}_l = \lambda_l \mathbf{v}_l, \quad ||\mathbf{v}_l|| = 1, \quad l = 0, 1, ..., n-1. \tag{2.4}$$

The set of eigenvectors $\{\mathbf{v}_l\}_{l=0}^{n-1}$ can be viewed as an orthonormal basis for $\mathcal{R}^n$, and each function $f \in \mathcal{R}^n$ can be expanded in terms of these eigenvectors.

The eigenvectors corresponding to the lowest eigenvalues of the graph Laplacian are the smoothest in the sense that $|\mathbf{v}_l(m) - \mathbf{v}_l(n)|$ is small for neighboring vertices m and n. At the extreme is $\mathbf{v}_0$ which is a constant vector, i.e. $\mathbf{v}_0(m) = \mathbf{v}_0(n)$ for all m, n. For more properties of the graph Laplacian eigenvectors, see [5].

2.3 Graph Fourier Transform

The Fourier transform in classical signal processing is a mathematical operation that decomposes a signal into its constituent frequencies in the Fourier domain. In the classical Fourier transform, the Fourier coefficients $\hat{f}$ are computed by the inner product between the function f and the exponential functions $e^{j\omega x}$ as follows.

$$\hat{f}(\omega) = \langle e^{j\omega x}, f \rangle = \int_{-\infty}^{+\infty} f(x) e^{-j\omega x} dx. \tag{2.5}$$

The inverse of the Fourier transform (2.6) is actually expanding the functions in terms of these exponential functions.

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat{f}(\omega) e^{j\omega x} d\omega, \tag{2.6}$$

where the set of exponential functions $\{e^{j\omega x}\}_{\omega \in \mathcal{R}}$ are the orthogonal basis for the space of squared integrable functions $L^2(\mathcal{R})$. They are actually the eigenfunctions of the first dimensional Laplacian operator, i.e. the second derivative operator as it is appeared in (2.7).

$$\frac{df}{dx^2} = -\omega^2 f. \tag{2.7}$$

For the functions defined on the vertices of a graph, there also exists a similar transform called the Graph Fourier Transform (GFT) [2]. Analogously to the classical Fourier transform in which we used eigenfunctions of one-dimensional Laplacian operator, i.e. exponential functions for the expansion of the squared integrable function $f \in \mathcal{R}$, we will use the eigenvectors of the graph Laplacian operator to define the GFT and inverse GFT for functions defined on graphs.

In order to compute the Fourier coefficients of a graph-function $f \in \mathcal{R}^n$, we just need to compute the inner product between the function and the eigenvectors of the graph Laplacian operator.

$$\hat{f}(l) = \langle \mathbf{v}_l, f \rangle = \sum_{k=1}^{n} \mathbf{v}_l^*(k) f(k). \tag{2.8}$$

The inverse of graph Fourier transform reads

$$f(k) = \sum_{l=0}^{n-1} \hat{f}(l) v_l(k). \tag{2.9}$$

In fact, it is the expansion of the function f in terms of the eigenvectors of graph Laplacian. Later, we will see how the graph Fourier transform can help us in constructing the Laplacian pyramid for multiscale analysis of graph-signals.

Chapter 3

The Classical Laplacian Pyramid

Multiscale data representation is one of the most useful idea in signal processing that particularly captures data in hierarchical structure. The Laplacian pyramid (LP) is one of the earliest such schemes proposed by Burt and Adelson [1]. It is especially useful for image coding. Multiresolution data representations are becoming increasingly popular especially in image processing applications. Pyramid data structures, in particular, play an important role in coding and are identically suited for progressive image transmission [6]. In this chapter we review the classical Laplacian pyramid framework.

3.1 Laplacian Pyramid Framework

The basic idea behind LP is first deriving a coarse approximation of the original signal by lowpass filtering and downsampling. Then, based on this coarse approximation, predict the original signal by upsampling followed by filtering and finally computes the error of prediction by comparing with original signal. In fact, LP can be viewed as an overcomplete transform that maps each signal to its coarse approximation and prediction error coefficients. In Fig. 3.1, a diagram for LP scheme is shown.

For special compression applications, the LP, which has a drawback of implicit oversampling, is normally replaced by subband coding or wavelet transforms which are critically sampled schemes.

With the notation as shown in Fig. 3.1 and writing signals as column vec-

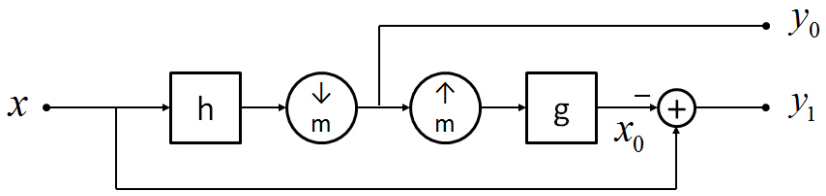

Figure 3.1: Analysis scheme in classical Laplacian pyramid.

tors, we can express the filtering followed by downsampling and upsampling followed by filtering operations as left matrix multiplications.

$$y_0 = \mathbf{H}x, \quad x_0 = \mathbf{G}y_0, \tag{3.1}$$

where $\mathbf{H}$ corresponds to h-filtering followed by downsampling by m and $\mathbf{G}$ corresponds to upsampling by m followed by g-filtering. It is easy to show that $\mathbf{H}$ has $\{\tilde{h}[n - mk]\}_{n \in Z}$ as its rows and $\mathbf{G}$ has $\{g[n - mk]\}_{n \in Z}$ as its columns, where $\tilde{h}[n] = h[-n]$.

The output of highpass channel in the LP is the prediction error

$$y_1 = x - x_0 = x - \mathbf{GH}x = (\mathbf{I} - \mathbf{GH})x, \tag{3.2}$$

where $\mathbf{I}$ is the identity operator with appropriate sizes depending of the context.

The analysis operator of the LP can be written as

$$\underbrace{\begin{pmatrix} y_0 \\ y_1 \end{pmatrix}}_{y} = \underbrace{\begin{pmatrix} \mathbf{H} \\ \mathbf{I} - \mathbf{GH} \end{pmatrix}}_{\mathbf{T_a}} x, \tag{3.3}$$

where $\mathbf{T_a}$ denotes the analysis operator and y denotes the transform coefficients. The usual inverse transform of the LP decomposition is the following.

$$\hat{x} = \underbrace{\begin{pmatrix} \mathbf{G} & \mathbf{I} \end{pmatrix}}_{\mathbf{T_s}} \underbrace{\begin{pmatrix} y_0 \\ y_1 \end{pmatrix}}_{y}, \tag{3.4}$$

where $\mathbf{T_s}$ denotes the synthesis operator and $\hat{x}$ is the reconstructed signal. Fig. 3.2 shows the usual synthesis scheme in the LP.

It is easy to check that $\mathbf{T_s T_a} = \mathbf{I}$ for any $\mathbf{H}, \mathbf{G}$. Thus, the LP is perfectly invertible. Since the LP is a redundant transform, its frame operator admits

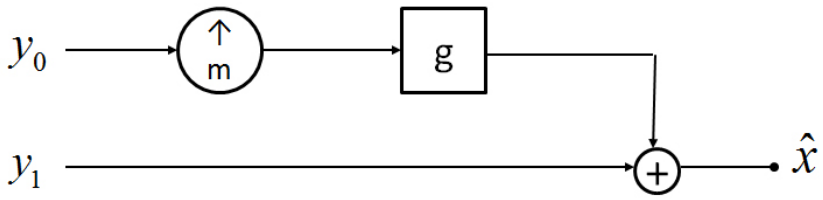

Figure 3.2: Usual synthesis scheme in classical Laplacian pyramid.

an infinite number of left inverses. The usual algorithm represented in (3.4) is just one of them. Among those, the most important one is the pseudo inverse of $\mathbf{T_a}$ denoted by $\mathbf{T_a}^\dagger$ as follows.

$$\mathbf{T_a}^\dagger = (\mathbf{T_a}^T\mathbf{T_a})^{-1}\mathbf{T_a}^T. \tag{3.5}$$

The importance of the pseudo inverse is its ability in eliminating the influence of additive noises in the frame coefficients as errors which are orthogonal to the range of the frame operator. In fact, if instead of having access to the frame coefficients $y = \mathbf{T_a}x$, we have $\hat{y} = y + e$, then the pseudo inverse provides the solution $\hat{x} = \mathbf{T_a}^\dagger\hat{y}$ that minimizes the residual $||\mathbf{T_a}\hat{x} - \hat{y}||_2$ [7]. This is actually what we call least-square solution which is very useful especially for denoising.

The reconstruction error can be easily computed for each left inverse denoted by $\mathbf{T_a}^{-1}$ as follows.

$$\begin{aligned} \hat{x} - x &= \mathbf{T_a}^{-1}\hat{y} - x \\ &= \mathbf{T_a}^{-1}(y + e) - x \\ &= \mathbf{T_a}^{-1}(\mathbf{T_a}x + e) - x \\ &= \mathbf{T_a}^{-1}e. \end{aligned} \tag{3.6}$$

When we replace the general left inverse $\mathbf{T_a}^{-1}$ with the pseudo inverse $\mathbf{T_a}^\dagger$ from (3.5), the reconstruction error becomes $(\mathbf{T_a}^T\mathbf{T_a})^{-1}\mathbf{T_a}^T e$, which is equal to zero for errors perpendicular to the range of the frame operator $\mathbf{T_a}$.

3.2 Tight Frame Case

In this section, we focus on the particular case where the analysis operator is a tight frame. A frame operator is called tight if the L_2 norm of the

signal is preserved under the frame operation. For a detailed introduction to frames, see [8]. It can be shown that where the filters in LP are orthogonal with respect to the sampling by m, the analysis operator would be a tight frame. The orthogonal condition requires that first, g has to be orthogonal to its shifted versions by m, i.e. $\langle g[.], g[.-mn]\rangle = \delta[n]$ and second, $h[n] = g[-n]$ or equivalently $\mathbf{H} = \mathbf{G}^T$. The Pythagorean theorem can easily be used in order to prove that the Laplacian pyramid with orthogonal filters is a tight frame [7].

Since $\mathbf{T_a}$ is a tight frame with ${\mathbf{T_a}}^T\mathbf{T_a} = \mathbf{I}$, its pseudo inverse is simply the transposed matrix ${\mathbf{T_a}}^T$. Thus we have

$$\mathbf{T_a}^\dagger = \begin{pmatrix} \mathbf{G}^T \\ \mathbf{I} - \mathbf{G}\mathbf{G}^T \end{pmatrix}^T = \begin{pmatrix} \mathbf{G} & \mathbf{I} - \mathbf{G}\mathbf{G}^T \end{pmatrix}. \tag{3.7}$$

So, the optimal reconstruction, i.e. the least-square solution using the pseudo inverse is

$$\hat{x} = \mathbf{T_a}^\dagger y = \mathbf{G}y_0 + (\mathbf{I} - \mathbf{G}\mathbf{G}^T)y_1 = \mathbf{G}(y_0 - \mathbf{H}y_1) + y_1. \tag{3.8}$$

The last expression above is derived to reduce the computational complexity of the pseudo inverse $\mathbf{T_a}^\dagger$. It leads to an efficient filter bank reconstruction of the LP that is shown in Fig. 3.3. Notice that the pseudo inverse in this case has a symmetrical structure with the forward transform, and hence it has the same order of complexity.

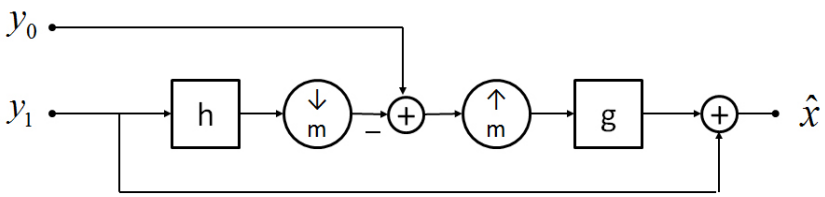

Figure 3.3: The optimum reconstruction scheme in classical LP, if the analysis operator is a tight frame.

For complexity reasons, if we restrict ourself to consider just the cases where the inverse operator has a fast structured transform as depicted in Fig. 3.3, we can also show that this structure could be an inverse transform of the LP on condition that the filters $\mathbf{H}$ and $\mathbf{G}$ are biorthogonal with respect to the sampling operator by m [7].

Recall that given a Hilbert space $\mathcal{H}$, a linear operator P mapping $\mathcal{H}$ onto itself is called a projector if $P^2 = P$. Furthermore, if P is selfadjoint or $P^T = P$ then P is called an orthogonal projector.

Biorthogonality of the filters $\mathbf{H}$ and $\mathbf{G}$, mentioned above, means that $\mathbf{GH}$ is a projector, i.e. $(\mathbf{GH})^2 = \mathbf{GH}$. So, the diagram depicted in Fig. 3.3 not only works for orthogonal filters ($\mathbf{H} = \mathbf{G}^T$); but, it also works for biorthogonal ones. In fact, if $\mathbf{G}$, $\mathbf{H}$ are such that $(\mathbf{GH})^2 = \mathbf{GH}$, then this diagram is the inverse transform of the LP.

Furthermore, it can be shown that the suggested reconstruction method in Fig. 3.3 is also a pseudo inverse if and only if $\mathbf{GH}$ is an orthogonal projector, i.e. $(\mathbf{GH})^2 = (\mathbf{GH})$ and $(\mathbf{GH})^T = \mathbf{GH}$ as well.

Chapter 4

Preliminary Tools For the Graph Laplacian Pyramid

In previous chapter, we saw the classical Laplacian pyramid scheme. We discussed the analysis and synthesis operators in LP framework and how we could implement them in effective ways. The goal of this chapter is to prepare the preliminary tools for developing the same ideas behind the classical LP for construction of the LP on the space of the functions defined on graphs, i.e. graph-signals. The graph Laplacian pyramid scheme will be discussed in the next two chapters.

Recall that in the classical LP scheme, a coarse version of the original signal is obtained by lowpass filtering followed by downsampling. Then, based on this coarse version of signal, the original one is predicted by upsampling of the coarse version followed by filtering. Finally the prediction error is computed. The first step to develop these ideas for LP decomposition of graph-signals is to define the filtering, downsampling, and upsampling concepts on graphs.

4.1 Filtering on Graphs

Filtering in the space of functions defined on vertices of a graph can be defined by a *graph Fourier multiplier operator* that reshapes the function's frequencies through multiplication in the Fourier domain.

For a function f defined on the real line, the Fourier multiplier operator Ψ acts as

$$\widehat{\Psi f}(\omega) = g(\omega)\hat{f}(\omega), \tag{4.1}$$

where g is referred as *multiplier* which is a real-valued function. Equivalently, we can write this as

$$\begin{aligned} \Psi f(x) &= \mathcal{F}^{-1}(\ g(\omega)\mathcal{F}(f)(\omega)\ )(x) \\ &= \frac{1}{2\pi}\int_{\mathcal{R}} g(\omega)\hat{f}(\omega)e^{j\omega x}d\omega, \end{aligned} \tag{4.2}$$

where $\mathcal{F}$ is the Fourier transform and $\mathcal{F}^{-1}$ is the inverse Fourier transform. The idea behind the Fourier multiplier operator Ψ is the following. Every squared integrable function f on a real line can be represented as a superposition of the exponential functions in the Fourier domain. The Fourier multiplier operator can modify f by changing the weight of contribution for each of the exponential function.

We can also extend this straightforwardly to the functions defined on the vertices of a graph as follows.

$$\begin{aligned} \Psi f(n) &= \mathcal{F}^{-1}(\ g(\lambda_l)\mathcal{F}(f)(l)\ )(n) \\ &= \sum_{l=0}^{n-1} g(\lambda_l)\hat{f}(l)\mathbf{v}_l(n), \end{aligned} \tag{4.3}$$

where the notations defined as before. In spectral graph theory, we know that the inverse graph Fourier transform (2.9) provides a representation of a signal f as a superposition of the orthonormal set of eigenvectors of the graph Laplacian. Analogously to the classical Fourier multiplier, the effect of the graph Fourier multiplier is actually to modify the contribution of each eigenvector. In fact, using a graph Fourier multiplier would be equivalent to going into the Fourier domain, change the components wisely and then coming back to the original domain.

As an extreme example, applying a step function as multiplier g that is 1 for all λ_l below some threshold, and 0 for all λ_l above the threshold is equivalent to the projection of the signal on the subspace spanned by eigenvectors associated with the lowest k eigenvectors, for some k chosen by the threshold. This is analogous to the lowpass filtering in the continuous domain.

Now, consider an undirected weighted graph G with $\mathbf{V} = [\mathbf{v}_0|\mathbf{v}_1|...|\mathbf{v}_{n-1}]$ as the matrix containing the eigenvectors of its graph Laplacian in columns. The graph Fourier coefficients $\hat{f}$ of a function f can be easily computed by $\hat{f} = \mathbf{V}^T f$, i.e. by computing the inner products of the eigenvectors and the function. Now, consider the multiplier function $h : \mathcal{R}^+ \to \mathcal{R}^+$. In matrix notation, we can write the graph Fourier multiplier operator $\mathbf{H}$ as

$$\mathbf{H}f = \mathbf{V}\tilde{\mathbf{H}}\mathbf{V}^T f, \tag{4.4}$$

where $\tilde{\mathbf{H}}$ is a diagonal matrix with on-diagonal entries $\{h(\lambda_l)\}_{l=0}^{n-1}$ and off-diagonal entries equal to zero.

Note that $\hat{f}_{mod} = \tilde{\mathbf{H}}\hat{f} = \tilde{\mathbf{H}}\mathbf{V}^T f$ are the modified graph Fourier coefficients and $\mathbf{H}f = \mathbf{V}\hat{f}_{mod}$ is the expansion of the modified function in terms of the eigenvectors of the graph Laplacian.

In order to see more precisely how a graph-signal can be filtered, we propose the following toy example. We consider 1000 vertices places randomly in the $[0, 1] \times [0, 1]$ square. We construct the weighted graph based on the threshold Gaussian kernel weighting function

$$\omega_{u,v} = \begin{cases} e^{-\frac{-|x_u - x_v|^2}{2\sigma^2}} & \text{if } ||x_u - x_v|| \leq \kappa \\ 0 & \text{otherwise} \end{cases}. \tag{4.5}$$

We let $\kappa = 0.075$ and $\sigma = 0.074$. The graph-signal shown in Fig. 4.1 is given by

$$f(i) = \begin{cases} x_{i,1}^2 + x_{i,2}^2 & \text{if } x_{i,2} < 1 - x_{i,1} \\ -2x_{i,1} & \text{otherwise} \end{cases}. \tag{4.6}$$

Here, $x_{i,1}$ and $x_{i,2}$ are the coordinates of x_i in the square. In the Voronoi diagram shown in Fig. 4.1, each cell corresponds to one vertex and the color of each cell indicates the value of graph-signal at that node.

Fig. 4.2 shows the filtered version of the original signal depicted in Fig. 4.1 when we use the following real-valued multiplier.

$$h(\lambda) = e^{-t\lambda}, \tag{4.7}$$

for $t = 3$. In this figure, it can be easily seen that the lowpass filtered graph-signal is smoothed and the discontinuity along the diagonal direction in the original graph-signal is removed.

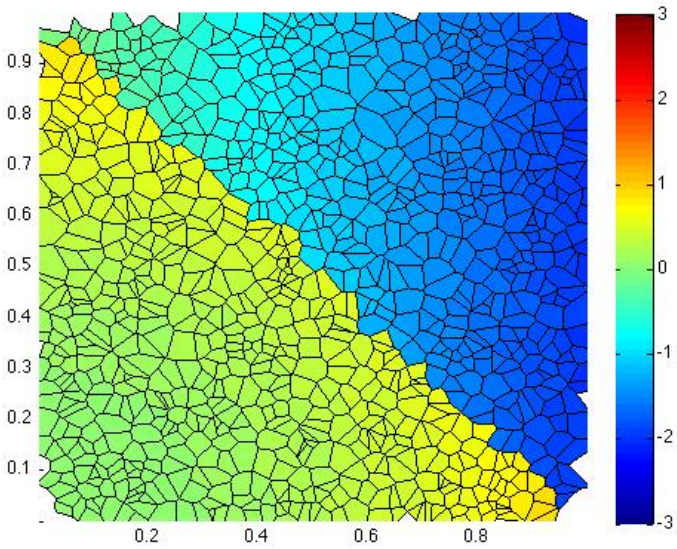

Figure 4.1: An example of piecewise smooth graph-signal with a discontinuity along $x_2 = 1 - x_1$.

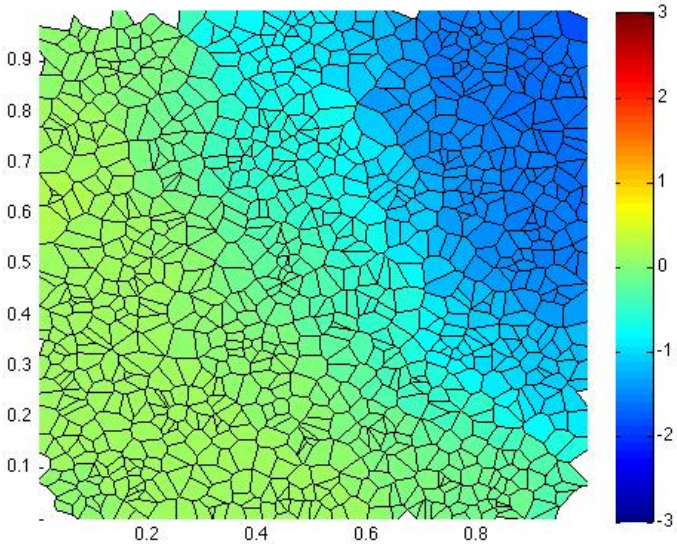

Figure 4.2: The lowpass filtered version of the graph-signal depicted in Fig. 4.1.

The difference between the original function and the lowpass filtered one is also shown in Fig. 4.3. It can be viewed as the output of a highpass channel. The discontinuity of the original graph-signal is easily recognizable in Fig. 4.3.

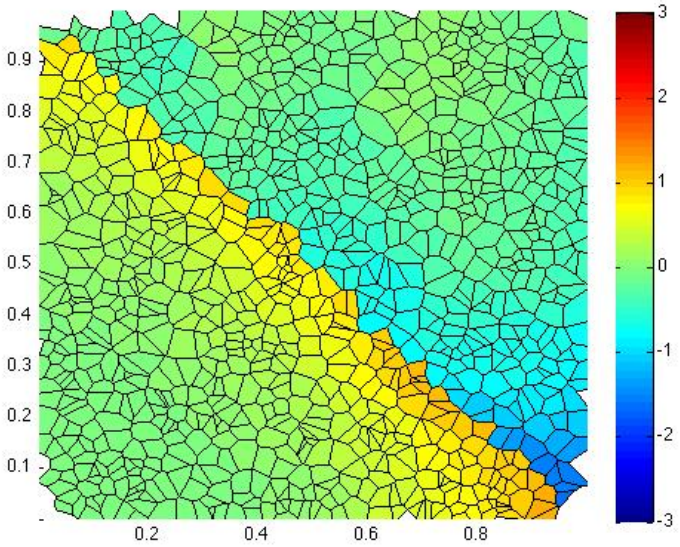

Figure 4.3: The difference between the original graph-signal and its lowpass filtered version. This signal can be viewed as the output of a highpass channel.

4.2 Downsampling and Upsampling on Graphs

In traditional signal processing applications, downsampling and upsampling are integral parts of critically sampled multi-rate filterbanks. In the classical LP framework Fig. 3.1, they are well-defined. However, in order to construct the LP for signals defined on graphs, we need to define downsampling and upsampling on graphs.

In classical signal processing, regular downsampling by m of a discrete function on a real line is defined by taking one out of m signal points. Similarly for signals on graphs, downsampling can be defined as keeping just a portion of entries of each function. We know that a graph-signal f can be viewed as an n-tuple vector where n is the number of vertices in graph. Downsampled version of a graph-function can be defined as a function whose

entries are a subset of the original function f. So, we need to have a method for vertex selection. The question is that how we could choose the best vertices for this aim? Vertex selection must be done in such a way that it keeps the structure of the original graph where the non-selected vertices are removed.

We know that the datasets on graphs can be defined as graph-signals which have a spectral interpretation given by eigenvalues and eigenvectors of the graph Laplacian matrix. As we saw earlier in spectral characteristics of weighted graphs, the eigenvectors corresponding to the smallest eigenvalues of graph Laplacian are the smoothest ones in the sense that the neighboring vertices have close values to each other at those eigenvectors. As an extreme example, the eigenvector corresponds to the smallest eigenvalue in a connected graph is a constant vector, the smoothest possible case.

Among all eigenvectors, the one corresponding to the largest eigenvalue of graph Laplacian is of interest to us. This eigenvector is somehow the non-smoothest one compared to its counterparts.

Our suggestion for choosing a subset of vertices for downsampling operation on graph is based on the sign of the eigenvector corresponding to the largest eigenvalue. This eigenvector is the least smooth amongst all eigenvectors; that is, the values of the components vary more rapidly across vertices connected by an edge. As a result, the sign of the components also varies, and the nodes with a positive component of the largest eigenvector are distributed through the graph.

Figs. 4.4-4.7 show the signs of some eigenvectors corresponding to the different eigenvalues of the graph Laplacian for the toy example mentioned in previous section. As you can see, the sign of the eigenvector corresponding to the largest eigenvalue Fig. 4.7 is more distributed than the other ones.

According to the sign of the largest eigenvector, we will divide vertices into two categories. Those with positive sign in the eigenvector associated to the largest eigenvalue are considered as selected ones and those with negative sign are considered as non-selected ones.

In these figures, positive and negative signs are distinguishable by different colors. This kind of vertex selection roughly keeps the structure of the original graph.

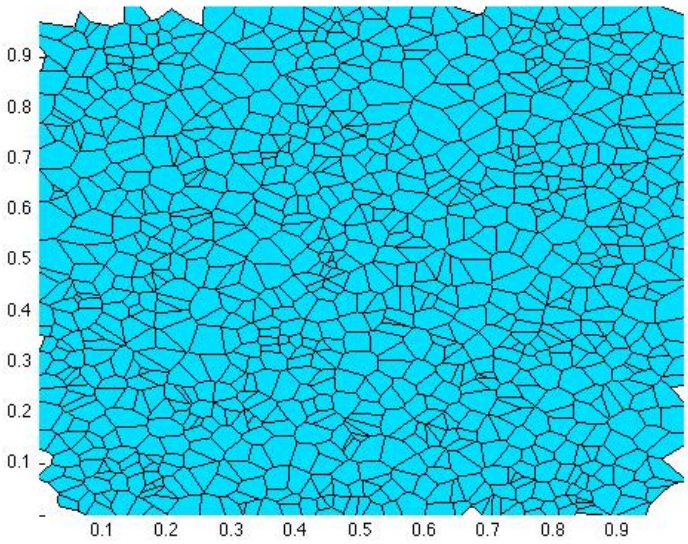

Figure 4.4: Sign of the eigenvector of a graph Laplacian matrix corresponding to λ_0.

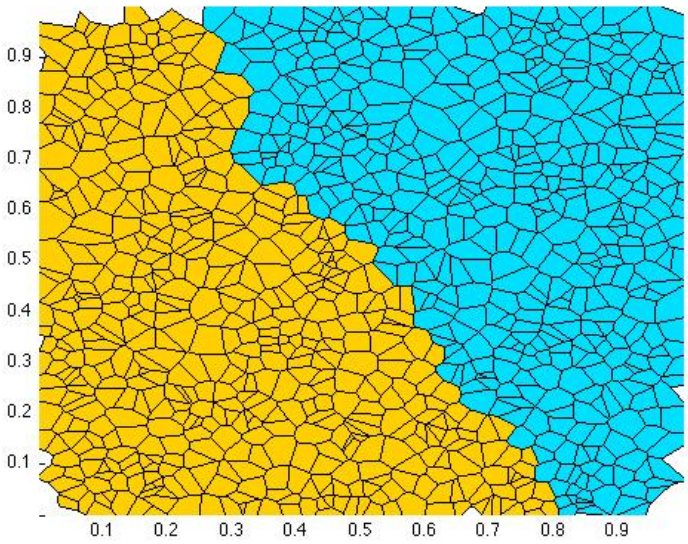

Figure 4.5: Sign of the eigenvector of a graph Laplacian matrix corresponding to λ_1.

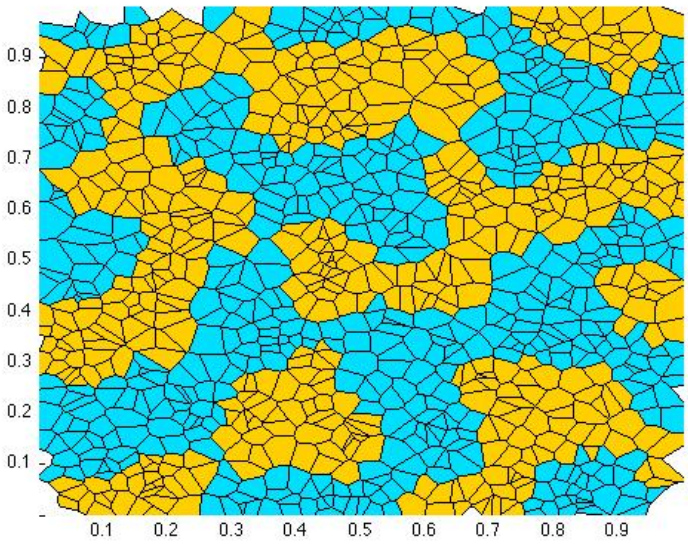

Figure 4.6: Sign of the eigenvector of a graph Laplacian matrix corresponding to λ_{49}.

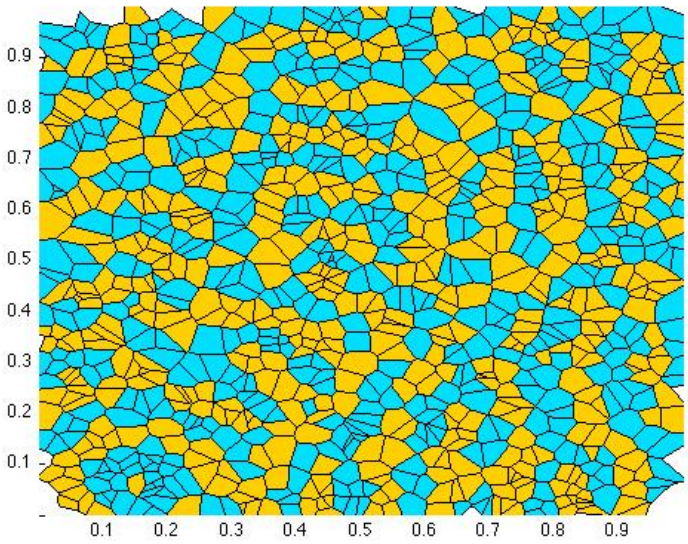

Figure 4.7: Sign of the eigenvector of a graph Laplacian matrix corresponding to $\lambda_{999} = \lambda_{max}$.

So, the downsampling of a graph-function f can be defined as choosing a subset $S \subset V$ such that all samples in f with indices not in S are discarded. Note that V is the set of all vertices in the graph and S contains the indices of vertices that have the positive sign in the eigenvector corresponding to the largest eigenvalue of the graph Laplacian matrix.

Upsampling is then defined by replacing zero for non-selected vertices in the function f. In other words, downsampling followed by upsampling is a masking operator where it masks the values of non-selected vertices. In matrix notation, $\mathbf{M}$ can be denoted as masking operator, i.e. downsampling followed by upsampling where it is an $n \times n$ diagonal matrix with on-diagonal entries correspond to the location of the selected vertices equal to one and zeros elsewhere.

In the following section, we focus on lattice structures to see how this method of vertex selection works.

4.3 Lattice Structures

In the previous section, we investigated how the eigenvector corresponds to the largest eigenvalue of the graph Laplacian matrix can help us to select an appropriate subset of vertices for downsampling. In this section, we look at some examples to see how this vertex selection method works on lattice structures.

An (n, k)-regular lattice is a graph with n nodes. Each node is connected to its k neighbors where $k = 2d$ is an even integer. Fig. 4.8 shows an example of regular lattice on a ring with 16 nodes connected to their 4 closest neighbors.

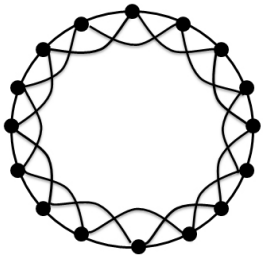

Figure 4.8: A regular lattice on a ring with $(n, k) = (16, 4)$.

The Laplacian matrix of such a graph is a circulant matrix

$$\mathcal{L} = \begin{bmatrix} c_0 & c_1 & c_2 & \dots & c_{n-1} \\ c_{n-1} & c_0 & c_1 & \dots & c_{n-2} \\ \dots & & & & \\ c_1 & c_2 & c_3 & \dots & c_0 \end{bmatrix}, \tag{4.8}$$

where each row consists of k ones and each row is the cyclic right shift of the row above. The coefficients of the first row are as follows.

$$\begin{aligned} c_0 &= k \\ c_p &= -1 \quad \forall 1 \leq p \leq d \\ c_{n-p} &= -1 \quad \forall 1 \leq p \leq d. \end{aligned} \tag{4.9}$$

Circulant matrices are special forms of Toeplitz matrices and their eigenvalues are the Discrete Fourier Transform (DFT) coefficients of their first rows [9]. By taking DFT from the first row of the Laplacian matrix (4.8) and taking into account the coefficients given in (4.9), we can easily compute the DFT coefficients of the first row which are equal to the eigenvalues of the graph Laplacian matrix.

$$\begin{aligned} C[l] &= \sum_{p=0}^{n-1} c_p e^{-j\frac{2\pi}{n}lp} \\ &= c_0 + \sum_{p=1}^{d} c_p e^{-j\frac{2\pi}{n}lp} + \sum_{p=n-d}^{n-1} c_p e^{-j\frac{2\pi}{n}lp} \\ &= c_0 + \sum_{p=1}^{d} c_p e^{-j\frac{2\pi}{n}lp} + \sum_{p=-d}^{-1} c_p e^{-j\frac{2\pi}{n}lp} \\ &= c_0 + \sum_{p=1}^{d} c_p e^{-j\frac{2\pi}{n}lp} + \sum_{p=1}^{d} c_p e^{+j\frac{2\pi}{n}lp} \\ &= c_0 + \sum_{p=1}^{d} c_p (e^{-j\frac{2\pi}{n}lp} + e^{+j\frac{2\pi}{n}lp}) \\ &= c_0 + \sum_{p=1}^{d} 2c_p \cos(\frac{2\pi}{n}lp) \\ &= k - 2\sum_{p=1}^{d} \cos(\frac{2\pi}{n}lp). \end{aligned} \tag{4.10}$$

Note that the set of eigenvalues $\{\lambda_l\}_{l=0}^{n-1}$ is equal to the set of DFT coefficients $\{C[l]\}_{l=0}^{n-1}$; but, λ_l is not equal to $C[l]$ necessarily because in our notation, the indexed eigenvalues are sorted, while it is not the case for the DFT coefficients $\{C[l]\}_{l=0}^{n-1}$. We denote the eigenvector corresponds to λ_l with $\mathbf{v}_l$ which is a column vector containing exponential terms as follows.

$$\mathbf{v}_l = [e^{-j\frac{2\pi}{n}lp}]_{n\times 1}, \quad p = 0, 1, ..., n-1. \tag{4.11}$$

Now, in the simplest case, consider an $(n, 2)$-regular lattice, i.e. n nodes placed on a ring where each node is connected to its two neighbors. It is actually equivalent to the one dimensional lattice ($G = \mathbf{Z}$) where n nodes are placed on a horizontal line and the boundary nodes, i.e. the first and the last ones on the line are connected as well, just to make the graph regular. In such a situation, the eigenvalues of the graph Laplacian matrix (4.10) would be

$$\{\lambda_l\}_{l=0}^{n-1} = \{2 - 2\cos(\frac{2\pi}{n}l)\}_{l=0}^{n-1}. \tag{4.12}$$

According to (4.12) and by assumption that the number of nodes n is an even integer, we will have

$$\lambda_{min} = \lambda_0 = C[0] = 0, \tag{4.13}$$

$$\lambda_{max} = \lambda_{n-1} = C[\frac{n}{2}] = 4. \tag{4.14}$$

The eigenvector $\mathbf{v}_{max}$ corresponds to the largest eigenvalue λ_{max} can be easily computed by

$$\begin{aligned} \mathbf{v}_{max} &= \mathbf{v}_{\frac{n}{2}} \\ &= [e^{j\frac{2\pi}{n}\cdot\frac{n}{2}\cdot p}]_{p=0}^{n-1} \\ &= [e^{j\pi p}]_{p=0}^{n-1} \\ &= [(-1)^p]_{p=0}^{n-1}. \end{aligned} \tag{4.15}$$

Let us come back to the method that we introduced for vertex selection. This method, which is based on the sign of the largest eigenvector $\mathbf{v}_{max}$, chooses one out of two nodes in a regular way because of the periodic change of the sign of $\mathbf{v}_{max}$ (4.15). In fact, it is exactly equivalent to the classical downsampling by two on a real line. Fig. 4.9 (a) depicts a one-dimensional lattice. Note that the boundary modification should be done by considering an extra edge between the first and the last nodes in order to make the

graph regular. Fig. 4.9 (b) shows how the vertices are selected for downsampling. Selected and non-selected vertices are identified by blue and red colors respectively.

Figure 4.9: Vertex selection in one-dimensional lattice.

Now consider a two-dimensional lattice structure ($G = \mathbf{Z}^2$) shown in Fig. 4.10 (a). Each non-boundary node is connected to its four closest neighbors. However, due to the finite nature of graph, the boundary nodes have some missing neighbors and hence the graph is not regular.

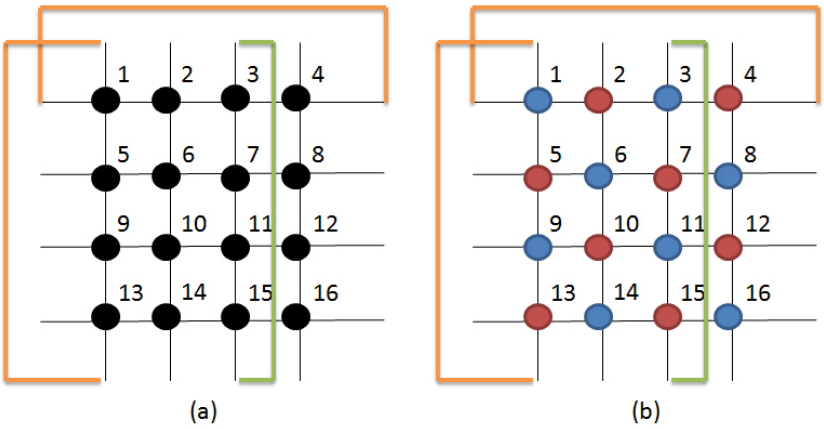

Figure 4.10: Vertex selection in two-dimensional lattice with $(n, k) = (16, 4)$.

In order to make the graph as $(n, 4)$-regular where $n = (2p)^2$ is squared of an even number, we propose to add extra edges between boundary nodes as follows. Look at the two-dimensional graph shown in Fig. 4.10 (a) as a matrix structure. All boundary nodes, except four of them placed at the corners, i.e. nodes with numbers $\{1, 4, 13, 16\}$, are already connected to their three closest neighbors. For example, the node number 3 is connected to the nodes with numbers $\{2, 4, 7\}$. In order to make the graph regular, we just need to add symmetry to the graph just by connecting those boundary nodes which are placed at the same row or column. For example, the node placed at coordinate $(1, 3)$, i.e. the node number 3 would be connected to the node at coordinate $(\sqrt{n}, 3)$, i.e. the node number 15. For those nodes placed at

the corners (four nodes mentioned above), we do the same except we have to consider symmetry in both horizontal and vertical directions. In fact, we have to connect them to two other nodes. For example, we need to connect the node placed at coordinate $(1, 1)$, i.e. the node number 1 to both nodes at coordinates $(1, \sqrt{n})$ and $(\sqrt{n}, 1)$, i.e. the nodes with numbers $\{4, 13\}$ and do the same for other nodes at other corners.

After boundary modification, we will construct the graph Laplacian matrix $\mathcal{L}$. Unlike the graph Laplacian matrix of the one-dimensional lattice which is circulant, the graph Laplacian matrix of the two-dimensional lattice is a little bit harder to formulate mathematically; but still it has a specific structure. For the boundary modified version of the two-dimensional lattice shown in Fig. 4.10 (a) which is a $(16, 4)$-regular graph, we have the following graph Laplacian matrix $\mathcal{L}$.

$$\mathcal{L} = \begin{bmatrix} \mathbf{\Phi} & -\mathbf{I_4} & \mathbf{0} & -\mathbf{I_4} \\ -\mathbf{I_4} & \mathbf{\Phi} & -\mathbf{I_4} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I_4} & \mathbf{\Phi} & -\mathbf{I_4} \\ -\mathbf{I_4} & \mathbf{0} & -\mathbf{I_4} & \mathbf{\Phi} \end{bmatrix}, \tag{4.16}$$

where $\mathbf{I_4}$ and $\mathbf{0}$ are the 4×4 identity and zero matrix respectively, and $\mathbf{\Phi}$ is a circulant matrix as follows.

$$\mathbf{\Phi} = \begin{bmatrix} 4 & -1 & 0 & -1 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ -1 & 0 & -1 & 4 \end{bmatrix}. \tag{4.17}$$

If we look at the eigenvector corresponding to the largest eigenvalue of the graph Laplacian matrix $\mathcal{L}$ in (4.16), and use the same method we discussed for vertex selection of one-dimensional lattice, we will have quincunx sampling as depicted in Fig. 4.10 (b). In this figure, blue nodes correspond to the vertices with positive sign in the largest eigenvector, i.e. the selected nodes, and the red ones are non-selected ones.

Even though someone can propose other way for vertex selection, corresponds to known cases like downsampling grids, the method based on the sign of the largest eigenvector is also a good way because it keeps the structure of the graph by selecting vertices which are distributed all over the graph.

4.4 A Toy Example in Filtering

In order to be more familiar with concept of filtering and downsampling in the space of the functions defined on graphs, we propose a simple problem and solve it. The problem is as follows. The constant input function f_{in}

$$f_{in} = \beta \mathbf{1}, \tag{4.18}$$

where $\mathbf{1}$ is an $n \times 1$ vector with all entries equal to 1 and $\beta \in \mathcal{R}$ is a constant real number, is passed through a lowpass filter h and then some of its elements will be masked, i.e. by downsampling followed by upsampling operators. If we pass the masked function through a second filter g, what should g be in order to have an output equal to the input function. In other words, what should g be to have perfect reconstruction.

Since the summation of the entries in each row of the graph Laplacian matrix $\mathcal{L}$ is equal to zero, we can easily see that the eigenvector $\mathbf{v}_0$ corresponds to the smallest eigenvalue $\lambda_0 = 0$ would be a constant vector. Moreover, since we consider the unit norm eigenvectors, we will have

$$\mathbf{v}_0 = \frac{1}{\sqrt{n}}\mathbf{1}. \tag{4.19}$$

Passing f_{in} through the filter h yields the output f_h in (4.20). Note that since both the input function f_{in} and the eigenvector $\mathbf{v}_0$ are constant vectors, and the set of $\{\mathbf{v}_l\}_{l=0}^{n-1}$ is an orthonormal basis for $\mathcal{R}^n$, we can easily conclude that f_{in} is perpendicular to the subspace spanned by $\{\mathbf{v}_l\}_{l=1}^{n-1}$, i.e. all the eigenvectors of graph Laplacian except $\mathbf{v}_0$.

$$\begin{aligned}
f_h &= \mathbf{V}\tilde{\mathbf{H}}\mathbf{V}^T f_{in} \\
&= \sum_{i=0}^{n-1} h(\lambda_i)\langle \mathbf{v}_i, f_{in}\rangle \mathbf{v}_i \\
&= h(\lambda_0)\langle \mathbf{v}_0, f_{in}\rangle \mathbf{v}_0 \\
&= h(\lambda_0)\beta\frac{1}{\sqrt{n}}\langle \mathbf{1}, \mathbf{1}\rangle \frac{1}{\sqrt{n}}\mathbf{1} \\
&= h(\lambda_0)\beta\frac{1}{\sqrt{n}}n\frac{1}{\sqrt{n}}\mathbf{1} \\
&= \beta h(\lambda_0)\mathbf{1}.
\end{aligned} \tag{4.20}$$

If $f_m = \mathbf{M}f_h$ denotes the masked version of f_h where $\mathbf{M}$ is the masking operator, the output of the second filter g is then computed as follows.

$$
\begin{aligned}
f_{out} &= \sum_{j=0}^{n-1} g(\lambda_j)\langle \mathbf{v}_j, f_m\rangle \mathbf{v}_j \\
&= \sum_{j=0}^{n-1} g(\lambda_j)\langle \mathbf{v}_j, \beta h(\lambda_0)\mathbf{M1}\rangle \mathbf{v}_j \\
&= \sum_{j=0}^{n-1} g(\lambda_j)\beta h(\lambda_0)\langle \mathbf{v}_j, \mathbf{M1}\rangle \mathbf{v}_j \\
&= \sum_{j=0}^{n-1} g(\lambda_j)\beta h(\lambda_0)(\sum_{p\in\mathcal{I}} v_{pj})\mathbf{v}_j \\
&= g(\lambda_0)\beta h(\lambda_0)(\sum_{p\in\mathcal{I}} v_{p0})\mathbf{v}_0 + \sum_{j=1}^{n-1} g(\lambda_j)\beta h(\lambda_0)(\sum_{p\in\mathcal{I}} v_{pj})\mathbf{v}_j \\
&= g(\lambda_0)h(\lambda_0)\beta r\frac{1}{\sqrt{n}}\frac{1}{\sqrt{n}}\mathbf{1} + \sum_{j=1}^{n-1} g(\lambda_j)\beta h(\lambda_0)(\sum_{p\in\mathcal{I}} v_{pj})\mathbf{v}_j \\
&= g(\lambda_0)h(\lambda_0)\frac{\beta}{\alpha}\mathbf{1} + \sum_{j=1}^{n-1} g(\lambda_j)\beta h(\lambda_0)(\sum_{p\in\mathcal{I}} v_{pj})\mathbf{v}_j, \qquad (4.21)
\end{aligned}
$$

where $\alpha = \frac{n}{r}$ is the downsampling factor, i.e. the ratio between the total number of elements and the number of non-masked entries, and $\mathcal{I}$ is the index set of non-masked elements. So $\sum_{p\in\mathcal{I}} v_{pj}$ indicates summation over all non-masked elements of $\mathbf{v}_j = [v_{pj}]_{p=1}^{n}$.

The first term in (4.21) is a constant vector. If we choose g such that

$$
g(\lambda_0) = \frac{\alpha}{h(\lambda_0)}, \qquad (4.22)
$$

then we will have

$$
f_{out} = f_{in} + \sum_{j=1}^{n-1} g(\lambda_j)h(\lambda_0)\beta(\sum_{p\in\mathcal{I}} v_{pj})\mathbf{v}_j. \qquad (4.23)
$$

We propose two options for g as follows. The first one is

$$
g(\lambda_j) = \frac{\alpha}{h(\lambda_j)}\delta_j. \qquad (4.24)
$$

In this case, $f_{in} = f_{out}$, i.e. we have perfect reconstruction because the second term in (4.23) would be equal to zero. The second choice of g is given by

$$g(x) = \frac{\alpha h(x)}{h(\lambda_0)^2}. \tag{4.25}$$

It is actually the normalized version of h. In this case, the error function $e = f_{out} - f_{in}$ would be

$$e = \sum_{j=1}^{n-1} \frac{\alpha h(\lambda_j)}{h(\lambda_0)} \beta(\sum_{p \in \mathcal{I}} v_{pj}) \mathbf{v}_j. \tag{4.26}$$

We know that h is a lowpass function and so it is a decreasing function. On the other hand, λ_0 is the smallest eigenvalue of the graph Laplacian. So, we can conclude that $\frac{h(\lambda_j)}{h(\lambda_0)} < 1$ for $j \neq 0$. Trivially, if we use sharper lowpass filters such as $h(x) = e^{-tx}$ with larger value of t, this ratio would be even smaller and the error function e will be decreased more.

Furthermore, by simulating different toy examples, we found out that $(\sum_{p \in \mathcal{I}} v_{pj})$ is around zero for $j \neq 0$. It means that even though we do not have the perfect reconstruction, the reconstruction error is low. As a result, choosing (4.25) for g yields not perfect, but roughly perfect reconstruction of the input signal.

Chapter 5

Single-Layer Graph Laplacian Pyramid

In the previous chapter, we became familiar with filtering, downsampling, and upsampling operators in graph-signals. Now, it is time to extend the idea behind the classical Laplacian pyramid to the space of functions defined on graphs. Recall that in the classical Laplacian pyramid, a coarse version of the original signal is obtained by lowpass filtering of the original signal followed by downsampling. Then, based on this coarse version, an approximate version of the original signal is reconstructed by upsampling followed by filtering. Finally, the reconstruction error is computed. In fact, the LP can be seen as a linear transform that maps a signal to its coarse and reconstruction error coefficients. Similar to what we defined for classical LP, we can design a Laplacian pyramid scheme for the space of functions defined on graphs as follows.

5.1 Graph LP Framework

In graph Laplacian pyramid, first the input graph-signal f would be passed through a lowpass filter h in order to get a coarse version of the input signal. After that, using the method discussed earlier for vertex selection, we downsample the function f by eliminating a portion of the entries in the smoothed version of the signal, i.e. the output of the filter h. Since we are interested in formulating the LP on graph-signals, we prefer to merge both downsampling and upsampling operators together to have a one single operator called the masking operator. In fact, since the upsampling is just putting zeros at the locations of non-selected vertices, we can model downsampling followed by

upsampling by masking operator $\mathbf{M}$ where $\mathbf{M}$ is a diagonal matrix with ones at on-diagonal entries correspond to the location of the selected vertices, and zeros elsewhere.

Then we will pass the output of the masking block through a second filter g in order to reconstruct the original function. Finally, the reconstruction error is easily computed by taking difference of the original signal and the output of the second filter.

Consider an input graph-signal $x \in \mathcal{R}^n$. In our notation, $y_0 = \mathbf{H_m}x$ denotes the output of h-filtering followed by masking operator. This is the output of the lowpass channel in the LP framework.

$$\begin{aligned} y_0 &= \mathbf{H_m}x \\ &= \mathbf{MH}x \\ &= \mathbf{MV}\tilde{\mathbf{H}}\mathbf{V^T}x, \end{aligned} \tag{5.1}$$

where $\mathbf{V} = [\mathbf{v}_0|\mathbf{v}_1|...|\mathbf{v}_{n-1}]$ is the matrix of the eigenvectors of graph Laplacian $\mathcal{L}$ and $\tilde{\mathbf{H}}$ is a diagonal matrix with on-diagonal entries $\{h(\lambda_l)\}_{l=0}^{n-1}$ and off-diagonal entries equal to zero. Recall that the multiplier is the real-valued function $h : \mathcal{R}^+ \to \mathcal{R}^+$.

The output of the highpass channel is then given by $y_1 = x - \mathbf{G}y_0$ which is equal to the reconstruction error.

$$\begin{aligned} y_1 &= x - \mathbf{G}x \\ &= x - \mathbf{V}\tilde{\mathbf{G}}\mathbf{V^T}x, \end{aligned} \tag{5.2}$$

where $\mathbf{V}$ is defined earlier and $\tilde{\mathbf{G}}$ is a diagonal matrix with on-diagonal entries $\{g(\lambda_l)\}_{l=0}^{n-1}$ and off-diagonal entries equal to zero. Note that for the second filter we use the multiplier $g : \mathcal{R}^+ \to \mathcal{R}^+$.

The analysis operator $\mathbf{T_a}$ is then defined in

$$\underbrace{\begin{pmatrix} y_0 \\ y_1 \end{pmatrix}}_{y} = \underbrace{\begin{pmatrix} \mathbf{H_m} \\ \mathbf{I - GH_m} \end{pmatrix}}_{\mathbf{T_a}} x, \tag{5.3}$$

where $y_0, y_1 \in \mathcal{R}^n$ are the coarse and prediction error coefficients respectively. Fig. 5.1 shows the analysis part of the graph Laplacian Pyramid.

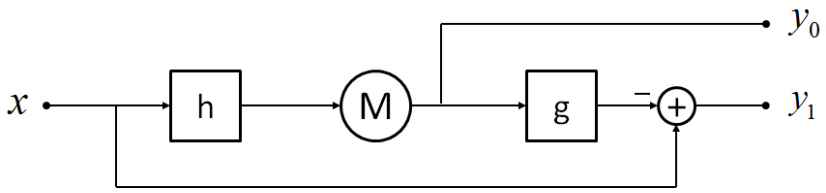

Figure 5.1: Analysis scheme in graph Laplacian pyramid.

The usual inverse transform of the LP for reconstruction of the original signal is also given in

$$\hat{x} = \underbrace{\begin{pmatrix} \mathbf{G} & \mathbf{I} \end{pmatrix}}_{\mathbf{T_s}} \underbrace{\begin{pmatrix} y_0 \\ y_1 \end{pmatrix}}_{y}. \tag{5.4}$$

First, we predict the original signal by filtering of the coarse version y_0 and add the reconstruction error y_1 to recover the original signal x completely. Fig. 5.2 shows the usual inverse transform of the graph LP.

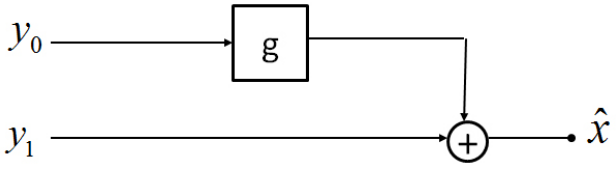

Figure 5.2: Usual synthesis scheme in graph Laplacian pyramid.

It is easy to check that $\mathbf{T_s}\mathbf{T_a} = \mathbf{I}$ for any $\mathbf{H_m}, \mathbf{G}$. In fact, it shows that LP can be perfectly reconstructed with any pairs of filters $\mathbf{H_m}, \mathbf{G}$. Analogously to the classical Laplacian pyramid, since the graph LP is also a redundant transform, an infinite number of left inverses are admitted as synthesis operator. The most important one among those is the pseudo inverse

$$\mathbf{T_a}^{\dagger} = (\mathbf{T_a}^T \mathbf{T_a})^{-1} \mathbf{T_a}^T. \tag{5.5}$$

As it is discussed previously in classical Laplacian pyramid, the importance of the pseudo inverse as a synthesis operator is its ability to eliminate the influence of those errors which are added to the transform coefficients y and are orthogonal to the range of the analysis operator $\mathbf{T_a}$. So, if instead of having access to $y = \mathbf{T_s}x$ we have $\hat{y} = y + e$, then the pseudo inverse provides the solution $\hat{x} = \mathbf{T_a}^{\dagger}\hat{y}$ that minimizes the residual $||\mathbf{T_a}\hat{x} - \hat{y}||_2$.

Considering (5.5) and (5.3), we can compute the pseudo inverse as follows.

$$\mathbf{T_a}^{\dagger} = [\mathbf{H_m}^T\mathbf{H_m} + (\mathbf{I} - \mathbf{GH_m})^T(\mathbf{I} - \mathbf{GH_m})]^{-1}(\ \mathbf{H_m}^T \quad \mathbf{I} - \mathbf{GH_m}^T\ ). \tag{5.6}$$

Regarding (5.6), it is maybe hard to compute the pseudo inverse explicitly. However, we can resolve this problem by using a gradient descent method to find the reconstructed signal iteratively instead of one-shot computation of the pseudo inverse. In the following chapter, we show how we can do this.

5.2 Implementation Issues

In this section, we consider two important implementation issues. One is the iterative solution of the reconstructed signal in synthesis part of the LP, instead of direct computation of the pseudo inverse, in order to mitigate the complexity of the computations. The other one is transferring the matrix computations to the one-dimensional real function computations by working in the spectral domain and using Chebyshev polynomials for approximating the multiplier functions h, g in filtering steps.

5.2.1 Iterative Solution of the Pseudo Inverse

We previously saw that the optimal reconstruction of the Laplacian pyramid is the pseudo inverse transform of the analysis operator. In order to see how the pseudo inverse is the least-square solution, we take derivative from the residual $||\mathbf{T_a}\hat{x} - \hat{y}||_2^2$ with respect to the analysis operator $\mathbf{T_a}$ as follows

$$\frac{d}{d\mathbf{T_a}}||\mathbf{T_a}\hat{x} - \hat{y}||_2^2 = 2\mathbf{T_a}^T(\mathbf{T_a}\hat{x} - \hat{y}), \tag{5.7}$$

and put (5.7) equal to zero. So,

$$\begin{aligned} \mathbf{T_a}^T(\mathbf{T_a}\hat{x} - \hat{y}) &= 0 \\ \rightarrow \mathbf{T_a}^T\mathbf{T_a}\hat{x} &= \mathbf{T_a}^T\hat{y} \\ \rightarrow \hat{x} &= \underbrace{(\mathbf{T_a}^T\mathbf{T_a})^{-1}\mathbf{T_a}^T}_{\mathbf{T_a}^{\dagger}}\hat{y}. \end{aligned} \tag{5.8}$$

Instead of computing the pseudo inverse explicitly, there is also an iterative way to solve (5.8) using a gradient descent method. In this method,

given transform coefficients y, we update the reconstructed signal iteratively as follows.

$$\hat{x}_{k+1} = \hat{x}_k + \tau \mathbf{T_a}^T (y - \mathbf{T_a}\hat{x}_k). \tag{5.9}$$

In (5.9), τ is the step size and has to be chosen sufficiently small for convergence condition. It is actually upperbonded by the inverse of the analysis operator norm [10]. $\hat{x}_k$ corresponds to the reconstructed signal at step k where $\hat{x}_0$ is a zero vector. After a sufficiently large number of iterations, we get the reconstructed signal which is slightly different from the original signal. The accuracy of the reconstruction improves with the number of iterations.

For implementation, we need to know what $\mathbf{T_a}^T$ and $\mathbf{T_a}^T\mathbf{T}$ are equivalent to because reforming (5.9), we have

$$x_{k+1} = (\mathbf{I} - \tau \mathbf{T_a}^T \mathbf{T_a}) x_k + \tau \mathbf{T_a}^T y. \tag{5.10}$$

In order to efficiently implement the iterative reconstruction of the graph-signal, it is better to know how the complementary operators $\mathbf{T_a}^T$ and $\mathbf{T_a}^T\mathbf{T_a}$ act. Regarding (5.3), we have the following.

$$\mathbf{T_a}^T = \begin{pmatrix} \mathbf{H_m}^T & \mathbf{I} - \mathbf{H_m}^T \mathbf{G}^T \end{pmatrix}, \tag{5.11}$$

$$\begin{aligned} \mathbf{T_a}^T \mathbf{T_a} &= \mathbf{H_m}^T \mathbf{H_m} + (\mathbf{I} - \mathbf{H_m}^T \mathbf{G}^T)(\mathbf{I} - \mathbf{G}\mathbf{H_m}) \\ &= \mathbf{H_m}^T \mathbf{H_m} + \mathbf{I} - \mathbf{G}\mathbf{H_m} - \mathbf{H_m}^T \mathbf{G}^T + \mathbf{H_m}^T \mathbf{G}^T \mathbf{G}\mathbf{H_m} \\ &= \mathbf{H_m}^T \mathbf{H_m} + \mathbf{I} - \mathbf{G}\mathbf{H_m} - \mathbf{H_m}^T \mathbf{G} + \mathbf{H_m}^T \mathbf{G}^2 \mathbf{H_m} \\ &= \mathbf{H_m}^T (\mathbf{I} + \mathbf{G}^2)\mathbf{H_m} + \mathbf{I} - (\mathbf{G}\mathbf{H_m} + \mathbf{H_m}^T \mathbf{G}). \end{aligned} \tag{5.12}$$

Recall that in our notation, $\mathbf{H_m}$ is the filtering followed by masking and so, $\mathbf{H_m}^T$ is the masking followed by filtering. In fact, $\mathbf{H_m}^T \neq \mathbf{H_m}$. However, since $\mathbf{G}$ is just filtering without any other operation, we can easily conclude that $\mathbf{G}^T = \mathbf{G}$. So, $\mathbf{G}^T\mathbf{G} = \mathbf{G}^2$ used in (5.12).

The diagrams depicted in Fig. 5.3 and Fig. 5.4 show how $\mathbf{T_a}^T$ and $\mathbf{T_a}^T\mathbf{T}$ can be implemented respectively. According to (5.10), the iterative method of reconstructing signal is then given by the diagram depicted in Fig. 5.5.

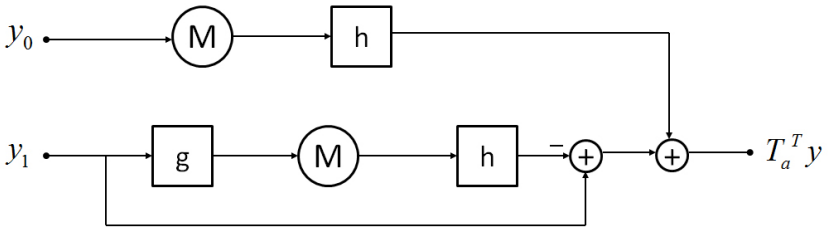

Figure 5.3: Complementary operator $\mathbf{T_a}^T$ for synthesis part of the graph LP.

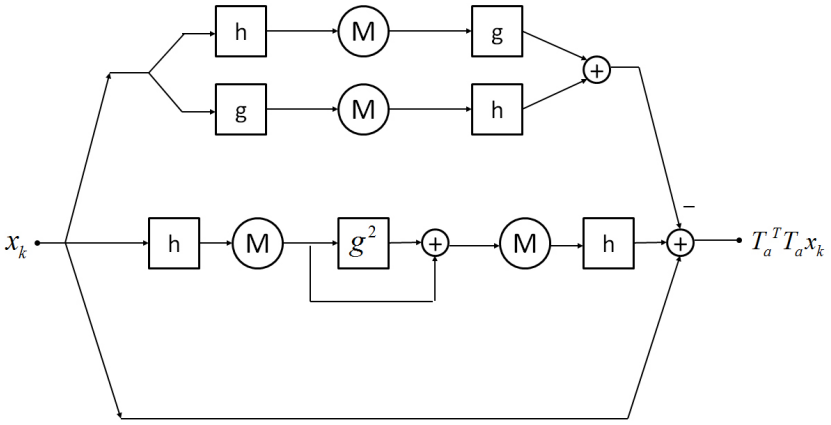

Figure 5.4: Complementary operator $\mathbf{T_a}^T\mathbf{T}$ for synthesis part of the graph LP.

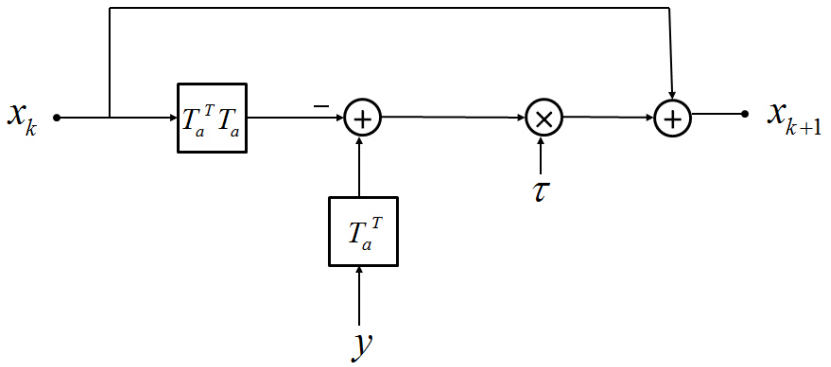

Figure 5.5: Iterative reconstruction of the graph-signal using gradient descent method.

5.2.2 The Chebyshev Polynomial Approximation

In the previous section, we discussed one important implementation issue in synthesis part of the Laplacian pyramid scheme which was about computation of the pseudo inverse operator. We proposed using gradient descent method instead of direct computation of the pseudo inverse. Even though iterative solution is a big step to get rid of complexity of direct computation, we can also make it even simpler by working in the spectral domain. In fact, we will see in the following that, using a simple trick, we can transfer all matrix computations to the one-dimensional real-valued computations. Furthermore, by using Chebyshev polynomials, we can also approximate the multiplier functions h and g used in filtering.

Considering iterative reconstruction of the signal in synthesis part of the graph Laplacian pyramid in (5.10), and starting with $x_0 = \mathbf{0}$, we can easily show that for any number N of iterations, x_N is

$$x_N = \tau \sum_{j=0}^{N-1} (\mathbf{I} - \tau\mathbf{Q})^j b, \tag{5.13}$$

where $\mathbf{Q} = {\mathbf{T_a}}^T \mathbf{T_a}$ is a real symmetric matrix and $b = {\mathbf{T_a}}^T y$ is a column vector.

In fact, in (5.10), $x_{k+1} = f(x_k)$ where $f(x) = (\mathbf{I} - \tau\mathbf{Q})x + \tau b$. The reconstructed signal at N^{th} iteration, i.e. x_N is what we look for and it is equal to $f_N(x_0) = \underbrace{f(f...f}_{N}(x_0)...)$. It is not hard to see that $f_N(x) = (\mathbf{I} - \tau\mathbf{Q})^N x + \sum_{j=0}^{N-1}(\mathbf{I} - \tau\mathbf{Q})^j \tau b$ and so, (5.13) is easily concluded because $x_N = f_N(x_0)$ for $x_0 = \mathbf{0}$.

Now, instead of matrix multiplications in Eq. (5.13), we can make it even simpler by working in the spectral domain.

Remember that the graph Fourier transform of a graph-function is given by the inner products of the function and the eigenvectors of graph Laplacian matrix $\mathcal{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$ where $\mathbf{\Lambda} = diag(\{\lambda_l\}_{l=0}^{n-1})$ and $\mathbf{V} = [\mathbf{v}_0|\mathbf{v}_1|...|\mathbf{v}_{n-1}]$ is the matrix consisting of the eigenvectors in columns. The l^{th} graph Fourier coefficient of a function $f \in \mathcal{R}^n$ is equal to $\hat{f}(l) = \langle \mathbf{v}_l, f \rangle$. So,

$$\begin{aligned}
\widehat{\mathcal{L}f}(l) &= \langle \mathbf{v}_l, \mathcal{L}f \rangle \\
&= \langle \mathbf{v}_l, \sum_{i=0}^{n-1} \lambda_i \mathbf{v}_i \mathbf{v}_i^T f \rangle \\
&= \langle \mathbf{v}_l, \sum_{i=0}^{n-1} \lambda_i \langle \mathbf{v}_i, f \rangle \mathbf{v}_i \rangle \\
&= \lambda_l \langle \mathbf{v}_l, f \rangle \\
&= \lambda_l \hat{f}(l), \qquad (5.14)
\end{aligned}$$

where the last equation comes from the fact that $\{\mathbf{v}_i\}_{i=0}^{n-1}$ is an orthonormal set. So, $\langle \mathbf{v}_i, \mathbf{v}_j \rangle = \delta(i-j)$.

Since $\mathbf{Q} = {\mathbf{T_a}}^T \mathbf{T_a}$ is a real symmetric matrix, it can also be written as $\mathbf{Q} = \mathbf{W}\mathbf{\Phi}\mathbf{W}^T$ for some $\mathbf{W} = [\mathbf{w}_0|\mathbf{w}_1|...|\mathbf{w}_{n-1}]$ and $\mathbf{\Phi} = diag(\{\phi_l\}_{l=0}^{n-1})$. Thus, $\mathbf{Q}$ can be seen as a graph Laplacian operator with orthonormal set $\{\mathbf{w}_l\}_{l=0}^{n-1}$ as eigenvectors and $\{\phi_l\}_{l=0}^{n-1}$ as non-negative eigenvalues. Using (5.14), we can easily show that $\widehat{\mathbf{Q}f}(l) = \phi_l \hat{f}(l)$ where $\hat{f}(l) = \langle \mathbf{w}_l, f \rangle$.

Since (5.13) is a polynomial with respect to $\mathbf{Q}$ and $\mathbf{Q}^j = \mathbf{W}\mathbf{\Phi}^j\mathbf{W}^T$, (5.13) can be transferred to the spectral domain as follows.

$$\widehat{x_N}(l) = \tau \sum_{j=0}^{N-1} (1 - \tau\phi_l)^j \hat{b}(l), \qquad (5.15)$$

where $\widehat{x_N}(l) = \langle \mathbf{w}_l, x_N \rangle$ and $\hat{b}(l) = \langle \mathbf{w}_l, b \rangle$ are the l^{th} graph Fourier coefficients x_N and b respectively. The reconstructed signal is then given by the inverse graph Fourier transform of $\widehat{x_N} = \sum_{l=0}^{n-1} \widehat{x_N}(l)\mathbf{w}_l$.

Since $\mathbf{Q}$ is a function of the analysis operator $\mathbf{T_a}$, it is a function of the multipliers h and g as well. Chebyshev polynomial approximation then can be used to simplify the computations more than what is expected. For more discussion about Chebyshev polynomial approximation, readers are referred to [11].

In the following we see the implementation results of the single-layer Laplacian pyramid with different types of reconstruction scheme mentioned in Section 5.1. The graph-signal is the same as what we defined earlier in the toy example of filtering. There are 1000 vertices distributed randomly in

$[0,1] \times [0,1]$. All parameters are the same except we used $h(\lambda) = e^{-0.1\lambda}$ and $g(\lambda) = e^{-\lambda}$ as multipliers and $\tau = 0.1$ as the step size in iterative reconstruction of the signal. Number of iterations is also considered as $N = 1000$.

Table. 5.1 shows the normalized mean squared errors of the reconstruction scheme. The standard deviation of the Gaussian noise which is added to the transform coefficients before the synthesis part of the LP is $\sigma_{noise} = 0.2$.

	Mean Squared Error
Usual inverse transform Eq. (5.4)	0.0388
Iterative pseudo inverse Eq. (5.10)	0.0369
Spectral reconstruction Eq. (5.15)	0.0369

Table 5.1: Reconstruction errors for different methods used in synthesis part of the graph LP. Standard deviation of the Gaussian noise is $\sigma_{noise} = 0.2$.

As we can see in Table 5.1, the mean squared errors of the iterative method of signal reconstruction using the pseudo inverse formula and the spectral reconstruction of the coefficients are the same. This is the case because they are actually equivalent to each other. One of them reconstructs the signal in the original domain, and the other one reconstructs it in the spectral domain. One more thing to note is that the mean squared error for the usual inverse transform is more than the iterative pseudo inverse because it is not the least-square solution, while the pseudo inverse is the optimum reconstruction scheme.

In Fig. 5.6, the reconstruction error function is shown where the iterative pseudo inverse scheme is used as synthesis part of the graph LP.

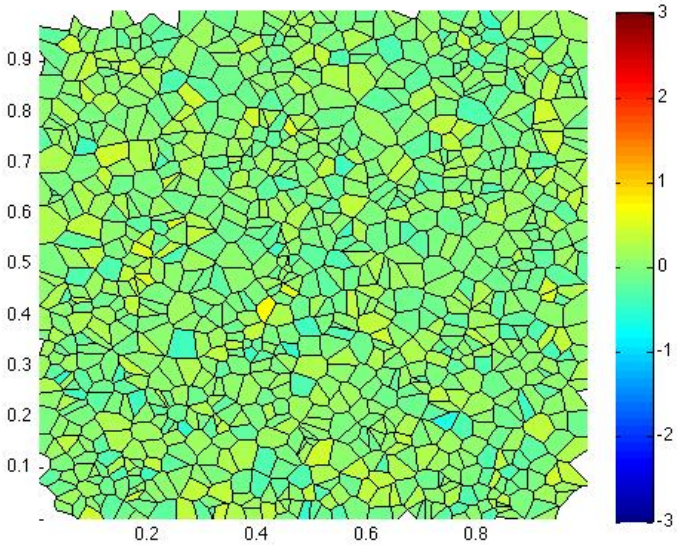

Figure 5.6: The reconstruction error where the iterative solution of the pseudo inverse is used as synthesis part of the graph LP. Standard deviation of the Gaussian noise is $\sigma_{noise} = 0.2$.

Chapter 6

Multi-Layer Graph Laplacian Pyramid

In the single-layer Laplacian pyramid scheme, the output of the analysis part consists of both coarse and prediction error coefficients. The synthesis part is then based on approximating the original signal via filtering of the coarse coefficients and adding the reconstruction errors to the output of the filter.

Similar to the classic multi-scale signal processing, we can consider the situation where the LP scheme is repeated on the coarse version to have multiscale representation of the graph-signal. In the following we take a look at multi-layer structure of the graph LP more precisely.

6.1 Multi-Layer Structure

We already know that the output of the analysis part of the single-layer LP is the coarse y_0 and prediction error y_1 coefficients. We can repeat the single-layer LP scheme for the lowpass branch by looking at y_0 as a new input signal instead of the original one x. In fact, y_0 can be viewed as an input signal in the lower dimensional space for the second layer of the LP scheme.

For example, consider a simple two-layer Laplacian pyramid. The input graph-signal is x and the output of lowpass and highpass branches of the LP in the first layer is y_0 and y_1 respectively. The second layer of LP would be considered on the lowepass branch of the first layer, i.e. where the output is y_0. Let us y_{00} and y_{01} denote the coarse and prediction error coefficients of

the LP scheme in the second layer respectively. The analysis operator maps graph-signal x to the coefficients $\{y_1, y_{00}, y_{01}\}$.

For the synthesis part, we will reconstruct the y_0 in the first step by using y_{00} and y_{01}, and then we could be able to reconstruct the original signal x using y_1.

In this new scheme, the output of the LP transform would be the whole set of reconstruction error coefficients in all layers, i.e. the outputs of high-pass channels, plus the coefficients correspond to the coarsest version of the original signal, i.e. the output of lowpass channel in the last LP layer. Perfect reconstruction is also guaranteed in the case of having access to these coefficients without any noise.

In order to construct the multi-layer Laplacian pyramid scheme, we need to be familiar with *graph reduction* techniques. Remember that filtering on the space of the functions defined on graphs is based on the spectral properties of the graph. In single-layer LP we use the eigenvectors and eigenvalues of graph Laplacian matrix; but, for multi-layer LP we need simplified versions of the original graph at each layer. The diagram depicted in Fig. 6.1 shows interior structure of the multi-layer Laplacian pyramid.

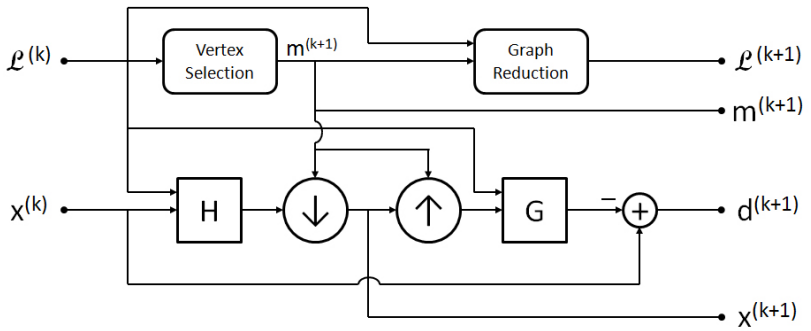

Figure 6.1: Multi-layer Laplacian pyramid scheme.

As we see in Fig. 6.1, the graph Laplacian matrix will be used for vertex selection and also graph reduction. The new graph Laplacian corresponding to the simplified graph is used for the next layer. At each layer, the coarse and detail coefficients are the outputs and the coarse coefficients can be regarded as new input for the next layer. Selected vertices used in masking operators, i.e. downsampling followed by upsampling are then transferred to

the next layer as well.

In the next section we will see the Kron reduction technique as a graph reduction method in use for multi-layer LP scheme.

6.2 Kron Reduction of Graphs

In many scientific applications, when you analyze data with network-like structure you will probably need to simplify the network in some sense. Manipulating with large matrices such as Laplacian matrices correspond to the networks could be so complex and also time consuming. Kron reduction is a mathematical method in use to simplify such networks. The Kron reduction of networks is ubiquitous especially in classical circuit theory and related applications in order to obtain lower dimensional electrically equivalent systems. Particularly, in the context of large-scale integration chips when you want to focus on behavior, synthesis, and analysis of resistive circuits you will probably need such an operation. Kron reduction is both a practically important and theoretically fascinating problem occurring in the reduction of the networks and their associated matrices. Some general applications of Kron reduction occur in sparse matrix algorithms, multi-grid solvers, finite element analysis, and Markov chains [12].

6.2.1 Preliminaries and Notations

Consider an undirected weighted graph G with adjacency and Laplacian matrices A and L respectively. Given a finite set $\mathcal{S}$, let $|\mathcal{S}|$ be its cardinality, and define for integer number n the index set $I_n = \{1, 2, ..., n\}$. We use the following standard notation for submatrices [13]: for two non-empty submatrices $\alpha, \beta \subseteq I_n$ let $A[\alpha, \beta]$ denote the submatrix of A obtained by the row indexed by α and the columns indexed by β and define the shorthands $A[\alpha, \beta) = A[\alpha, I_n \setminus \beta], A(\alpha, \beta] = A[I_n \setminus \alpha, \beta], A(\alpha, \beta) = A[I_n \setminus \alpha, I_n \setminus \beta]$ where $\setminus$ denotes the set difference. Let $\alpha \subseteq I_n$ be a proper subset of nodes with $|\alpha| > 2$. We define the $(|\alpha| \times |\alpha|)$ dimensional Kron-reduced matrix L_{red} by

$$L_{red} = L/L(\alpha, \alpha) = L[\alpha, \alpha] - L[\alpha, \alpha)L(\alpha, \alpha)^{\dagger}L(\alpha, \alpha], \tag{6.1}$$

where $\dagger$ is denoted for the pseudo inverse. In fact, Kron-reduced version of a graph is a graph whose Laplacian matrix is obtained by the Schur complement of the original Laplacian matrix with respect to a subset of nodes

[12]. Remember that in downsampling on graphs we divide the whole set of vertices into two subgroups: boundary nodes correspond to the selected vertices and the interior nodes correspond to the non-selected ones. Here α denotes the selected vertices, i.e. the boundary nodes. In other words, the goal is to eliminate the interior nodes $\beta = I_n \setminus \alpha$ through Kron reduction.

The Kron-reduced matrix $L_{red} = L \setminus L(\alpha, \alpha)$ is well defined in the sense that you can associate an undirected weighted graph to L_{red}. In the following we will focus on structural, topological, spectral, and algebraic properties of Kron reduction. For more precise details of Kron reduction and Schur complement operation, we refer readers to [12] and [13] respectively.

6.2.2 Properties of Kron Reduction

Two nodes $i, j \in \alpha$ are connected by an edge in G_{red} if and only if there is a path from i to j in G whose nodes all belong to $\{i, j\} \cup (I_n \setminus \alpha)$. Topological connectivity among the boundary nodes becomes only denser under Kron reduction. Hence, the algebraic connectivity which is equal to the smallest non-zero eigenvalue of Laplacian matrix λ_1, should increase accordingly. In particular, Kron reduction preserves connectivity; if the original graph is connected, its Kron reduction is also connected.

Another important feature of Kron reduction operation is spectral interlacing. For any $r \in \{0, 1, 2, ..., |\alpha| - 1\}$ the following holds

$$\lambda_r(L) \leq \lambda_r(L_{red}) \leq \lambda_r(L[\alpha, \alpha]) \leq \lambda_{r+n-|\alpha|}(L). \tag{6.2}$$

From an algebraic point of view, Kron reduction is closed under irreducibility. If L is irreducible, then L_{red} is also irreducible. Another important feature of Kron reduction concerns the monotonicity of the elements. For all $i, j \in \alpha$,

$$L_{red}(i, j) \leq L[\alpha, \alpha](i, j). \tag{6.3}$$

Equivalently, for all $i, j \in \alpha$,

$$A_{red}(i, j) \geq A[\alpha, \alpha](i, j). \tag{6.4}$$

The most important property of Kron reduction is that the effective resistance R_{ij} between any two boundary nodes $i, j \in \alpha$ is equal when computed from L or L_{red}. Effective resistance is measured by

$$R_{ij} = (e_i - e_j)^T L^\dagger (e_i - e_j), \tag{6.5}$$

where e_i is a column vector with zero entries except at location i.

In classical circuit theory, the current-balance equations $I = QV$ are obtained by Kirchhoff's and Ohm's lows, where I is the vector of injected currents at the nodes and V is the vector of the nodal voltages. Q is the electrical conductance matrix. When the Laplacian matrix L is replaced by conductance matrix Q, effective resistance R_{ij} would be exactly equivalent to the electrical resistance between nodes i, j. This property shows that the electrical resistance between boundary nodes are unchanged after Kron reduction. This is important for chip designers in order to simplify electrical networks while keeping effective resistance between boundary nodes unchanged.

6.2.3 Kron Reduction on a Lattice Structure

Here, we are interested in discovering the impact of the Kron reduction technique on a one-dimensional lattice structure ($G = \mathbf{Z}$) as an example. At the end, we will see that the Kron-reduced version of a one-dimensional lattice is also a one-dimensional lattice fewer nodes. In other words, the set of one-dimensional lattices is closed under Kron reduction operation.

We want to compute the Kron-reduced version of the one-dimensional lattice with an even number of nodes. The Schur complement must be done with respect to a subset of nodes. So, the first step is to choose a subset of nodes. In fact, we have to choose which vertices we are interested in selecting. This is exactly what we did for downsampling on graph.

We divide the whole set of vertices to two subsets with no common element in such a way that the union of these two subsets covers the whole set of vertices. One of them is called the interior nodes which will be eliminated through Kron reduction, and the other one is called the subset of boundary nodes, which includes the vertices of Kron-reduced graph.

The goal is to eliminate the interior nodes by means of Kron reduction operation. We choose the boundary nodes $\alpha \subseteq I_n$ just by looking at the sign of the largest eigenvector $\mathbf{v}_{max}$ corresponding to the largest eigenvalue of the graph Laplacian matrix L. Indices with positive signs in $\mathbf{v}_{max}$ are indicated

as boundary nodes and the rest are interior nodes.

We already know from (4.8) that a one-dimensional lattice has a circulant Laplacian matrix . Let us denote a circulant matrix with $\mathbf{Circ}\{c_0, c_1, c_2, ..., c_{n-1}\}$ where $\{c_0, c_1, c_2, ..., c_{n-1}\}$ is the first row of the matrix and each row is the cyclic right shift of the row above. So, for a one-dimensional lattice we have $L = \mathbf{Circ}\{2, -1, \underbrace{0, ..., 0}_{n-3}, -1\}$, where n is the number of vertices.

Regarding (4.15), we know that the boundary nodes subset chosen based on the sign of $\mathbf{v}_{max}$ in a one-dimensional lattice with an even number of nodes is $\alpha = \{1, 3, 5, ..., n-1\}$. It is also easy to see that for this case $L[\alpha, \alpha] = L(\alpha, \alpha) = 2I_{\frac{n}{2}}$ and $L[\alpha, \alpha) = L(\alpha, \alpha]^T = \mathbf{Circ}\{-1, \underbrace{0, ..., 0}_{\frac{n}{2}-2}, -1\}$, where n is assumed to be an even number and $I_{\frac{n}{2}}$ is $\frac{n}{2} \times \frac{n}{2}$ identity matrix. So, we have

$$\begin{aligned} L_{red} &= L[\alpha, \alpha] - L[\alpha, \alpha)L(\alpha, \alpha)^{\dagger}L(\alpha, \alpha] \\ &= 2I_{\frac{n}{2}} - \frac{1}{2}FF^T, \end{aligned} \tag{6.6}$$

where $F = L[\alpha, \alpha)$. It is not hard to see $FF^T = \mathbf{Circ}\{2, 1, \underbrace{0, ..., 0}_{\frac{n}{2}-3}, 1\}$ and so, by (6.6), $L_{red} = \mathbf{Circ}\{1, -0.5, \underbrace{0, ..., 0}_{\frac{n}{2}-3}, -0.5\}$.

Comparing L_{red} to $L = \mathbf{Circ}\{2, -1, \underbrace{0, ..., 0}_{n-3}, 1\}$, we can see that the Kron-reduced Laplacian matrix also corresponds to the Laplacian matrix of a one-dimensional lattice where the number of vertices is halved and the weights of edges are scaled by 0.5.

Chapter 7

Conclusions and Future Works

In this report, we have presented a framework for constructing the multi-layer Laplacian pyramid on the space of functions defined on graphs. We have shown how the graph Fourier multiplier operator can be used for filtering of graph-signals. We have also proposed a method for downsampling datasets defined on arbitrary weighted graphs. The results are especially useful for graph-data compression in many domains such as networks and meshes.

In other words, we have shown how we can extend the classical concepts in signal processing such as filtering, downsampling, and upsampling for the graph-signals based on the spectral characterization of the weighted graphs and the polarity of the largest eigenvector in order to split the set of vertices into two subsets for the downsampling.

We have also discussed about Kron reduction as a mathematical tool for simplifying the structure of graphs, which is indeed a vital step for multi-scale signal processing. We considered lattice structures as examples of weighted graphs to see the impact of the Kron reduction operation and vertex selection method we use for graph downsampling.

By analogy with classical Laplacian pyramid, we have shown that the optimum synthesis operator is the least square solution equivalent to the pseudo inverse of the analysis operator. Even though we can compute the pseudo inverse explicitly, it maybe computationally challenging to do so. So, we discussed two important implementation issues. One of them is to iteratively reconstruct the graph-signal instead of direct computation of the pseudo inverse. The other one is transferring matrix computations to the one-dimensional real-valued computations and also using Chebyshev polynomials for approximating multiplier functions.

In future work, we would like to introduce some other methods of down-sampling. We are also interested in defining biorthogonal filters in the LP structure on graphs. It is not so easy to design biorthogonal filters in the graph LP framework because the spectral behavior of the weighted graph must be considered in filtering.

There are many possible directions for future research for improving or extending multi-layer Laplacian pyramid. To find the relation between eigenvectors of the Kron-reduced graph versus the original one is also an important task that should be investigated more precisely in future researches. Furthermore, we would also like to think about properties like critically sampled vs redundant, invertibility, perfect reconstruction, ease of filter design to guarantee desirable outcomes more precisely.

Bibliography

[1] P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact image code," *IEEE Trans. Commun.*, vol. 31, no. 4, pp. 532-540, April 1983.

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

[3] S. K. Narang, A. Ortega, "Local two channel critically sampled filter-banks on graphs," *ICIP*, Sept. 2010.

[4] F. K. Chung, "Spectral graph theory," *CBMS Regional Conference Series in Mathematics*, vol. 92, AMS bookstore, 1997.

[5] T. Biyikoglu, J. Leydold, and P. F. Stadler, "Laplacian eigenvectors of graphs," *Lecture notes in Mathematics*, vol. 1915, Springer, 2007.

[6] M. Unser, "An improved least square Laplacian pyramid for image compression," *Signal Processing*, vol. 27, no. 2, pp. 187-203, May 1992.

[7] M. N. Do and M. Vetterli, "Frame reconstruction of the Laplacian pyramid," *ICASSP*, May 2001.

[8] I. Daubechies, "Ten lectures on wavelets," *SIAM*, Philadelphia, PA, 1992.

[9] R. Olfati-Saber, "Algebraic connectivity ratio of Ramanujan graphs," *American Control Conf.*, July 2007.

[10] H. W. Engl, M. Hanke, and A. Neubauer, "Regularization of Inverse Problems", Kluwer, Dordrecht, 1996.

[11] D. I. Shuman, P. Vandergheynst, and P. Frossard, "Chebyshev polynomial approximation for distributed signal processing," *DCOSS*, June 2011.

[12] F. Dorfler and F. Bullo, "Kron redunction of graphs with applications to electrical networks," arXiv:1102.2950v1.

[13] F. Zhang, "The Schur complement and its applications," Springer, 2005.