MULTI-GRAPH LEARNING OF SPECTRAL GRAPH DICTIONARIES

Dorina Thanou and Pascal Frossard

Signal Processing Laboratory (LTS4), EPFL, Switzerland
Email:{dorina.thanou, pascal.frossard}@epfl.ch

ABSTRACT

We study the problem of learning constitutive features for the effective representation of graph signals, which can be considered as observations collected on different graph topologies. We propose to learn graph atoms and build graph dictionaries that provide sparse representations for classes of signals, which share common spectral characteristics but reside on the vertices of different graphs. In particular, we concentrate on graph atoms that are constructed on polynomials of the graph Laplacian. Such a design permits to abstract from the precise graph topology and to design dictionaries that can be trained and eventually used on different graphs. We cast the dictionary learning problem as an alternating optimization problem where the dictionary and the sparse representations of training signals are updated iteratively. Experimental results on synthetic graph signals representing common processes on graphs show that our dictionaries are able to capture the important components in graph signals. Further experiments on traffic data confirm the benefits of our dictionaries in the sparse approximation of signals capturing traffic bottlenecks.

Index Terms— sparse approximations, graph signal processing

1. INTRODUCTION

Many data analysis and processing tasks nowadays have to handle large structured datasets, where the structure information actually carry important information about the nature of the observations or data measurements. Examples of such datasets can be found in numerous application domains, such as transport, social or computer networks, brain analysis or even images. Such data can be represented as graph signals, where function values on the graph vertices represent the observed data, and the connection between vertices denote the relationship between different observations. As a result, a graph signal can be considered as an observation of a process evolving on a particular network domain [1], where the actual signal value depends on both the causes of the underlying process and the graph topology.

When graphs signals mostly capture the effect of a few processes on a graph topology, they can be modelled as the linear combinations of a small number of constitutive components, which we call graph atoms. In practice however, these signals do not all live on the exact same graph topology. Still, different graphs signals on different topologies may share similar spectral characteristics in the case where the underlying processes are similar. For example, a heat diffusion process may generate different graphs signals for different graph topology instances, even if the diffusion process always follows the same model. In order to have effective signal representations, it becomes therefore important to design a learning strategy that can abstract from the exact graph topology of each graph signal, and identify the common causes in specific classes of graph signals.

Several representation methods have been proposed for signals living on graphs, but they are adapted to general classes of signals. The graph Fourier transform for example has been shown to sparsely represent smooth graph signals [2]. Wavelet transforms such as diffusion wavelets [3], spectral graph wavelets [4], and critically sampled two-channel wavelet filter banks [5] mostly target piecewise-smooth graph signals. Then, vertex-frequency frames [6] can be used to analyze signal content at specific vertex and frequency locations. However, these methods cannot be adapted to specific types of graph signals. Such adaptation is generally achieved by training a dictionary that can sparsely represent a particular class of signals. Unfortunately, existing algorithms for dictionary learning techniques for graph signals [7], [8] rely on the assumption that all the training signals lie on the same graph, which is pretty constraining in practice. Moreover, classical dictionary learning techniques such as K-SVD [9] cannot handle training signals with different dimensionality, thus they cannot be applied to our problem.

In this paper, we propose a dictionary learning algorithm for graph signals that possibly live on different weighted graphs. Given a class of graph signals that live on a set of weighted graphs, we want to construct an overcomplete dictionary of atoms that can sparsely represent graph signals as linear combinations of only a few atoms of the dictionary. We want the atoms to capture the common components in the different graph signals, even if they lie on different topologies. We focus on the spectral characteristics of such constitutive components and propose to learn atoms that are polynomial functions of the graph Laplacian. The polynomial coefficients therefore define a kernel that can generate signals on different graph topologies. We cast the dictionary learning problem as an alternating optimization problem, where the polynomial coefficients of the atoms and the sparse codes of the training signals are updated iteratively. We then perform experiments on graph signals that represent common processes on different graphs and show that our dictionary learning method is able to recover the core components of these signals. We finally confirm the performance of our algorithm on traffic data, where the learnt dictionary is shown to provide better sparse approximations than non-adaptive representations, or representations that are optimized on different graphs independently.

Finally, as our algorithm permits to learn an effective graph signal representation from training signals living on different graphs, it surely provides important benefits in addition to effective approximation performance. In practice, it is common to have graph signals that live on different topologies or on graphs that do no have exactly the same number of nodes. Then, it might be necessary to segment large graphs into smaller graphs for complexity reasons, or for augmenting the number of training data. In all these cases, our algorithm is helpful as it goes beyond the limitations of traditional learning solutions that require training signals living on a fixed topology.

2. MULTI-GRAPH DICTIONARY LEARNING

2.1. Representation of graph signals

We consider a weighted and undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$, where $\mathcal{V}$ and $\mathcal{E}$ represent the vertex and edge sets of the graph $\mathcal{G}$, and W represents the matrix of its positive edge weights, with $W(i,j)$ denoting the weight of an edge connecting vertices i and j, and D is the diagonal degree matrix whose i^{th} diagonal element is equal to the sum of the weights of all the edges incident to vertex i [10]. A graph signal y in the vertex domain is a real-valued function defined on the vertices of the graph $\mathcal{G}$, such that $y(v)$ is the value of the function at vertex $v \in \mathcal{V}$. The normalized graph Laplacian operator is defined as $\mathcal{L} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$, and it is a real symmetric matrix that has a complete set of orthonormal eigenvectors with corresponding nonnegative eigenvalues. We denote its eigenvectors by $\chi = [\chi^1, \chi^2, ..., \chi^N]$, and the spectrum of eigenvalues by $\Lambda := \Big\{0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_{(N-1)} \leq 2\Big\}$.

We consider a general class of graph signals that are linear combinations of (overlapping) graph patterns positioned at different vertices on the graph. Each pattern captures the local variation of the signal in the neighborhood of a vertex, and it can be considered as a function whose values in the neighborhood depend on the local connectivity around that vertex. We represent the translation of a pattern across the vertices of the graph [4], [6] through a graph operator defined as

$$\widehat{g}(\mathcal{L}) = \chi \widehat{g}(\Lambda) \chi^T, \tag{1}$$

where χ, Λ are the eigenvectors and eigenvalues of the graph Laplacian matrix. The generating kernel $\widehat{g}(\cdot)$, that is a function of the eigenvalues of the Laplacian, characterizes the graph pattern in the spectral domain. In particular, if the generating kernel is smooth, $\widehat{g}(\mathcal{L})$ consists of a set of N columns, each representing a localized atom, generated by the same kernel, and positioned on different nodes on the graph [4], [1]. One can thus design graph operators consisting of localized atoms in the vertex domain by taking the kernel $\widehat{g}(\cdot)$ in (1) to be a smooth polynomial function of degree K [4], [7]:

$$\widehat{g}(\lambda_\ell) = \sum_{k=0}^{K} \alpha_k \lambda_\ell^k, \quad \ell = 0, ..., N-1. \tag{2}$$

The corresponding operator is then defined as

$$\widehat{g}(\mathcal{L}) = \chi \left(\sum_{k=0}^{K} \alpha_k \Lambda^k \right) \chi^T = \sum_{k=0}^{K} \alpha_k \mathcal{L}^k. \tag{3}$$

Note that the atom given by column n is equal to the polynomial $\widehat{g}(\cdot)$ of order K translated to the vertex n. The polynomial structure of the kernel $\widehat{g}(\cdot)$ ensures that the resulting atom has its support contained in a K-hop neighborhood of vertex n [4]. A graph dictionary is then defined as a concatenation of different graph operators or subdictionaries. It takes the form $\mathcal{D} = [\widehat{g_1}(\mathcal{L}), \widehat{g_2}(\mathcal{L}), ..., \widehat{g_S}(\mathcal{L})]$, where each subdictionary captures a different graph pattern in the spectral domain that is translated across all the vertices of the graph. Finally, a graph signal y can be expressed as a linear combination of a set of atoms generated from different graph kernels $\{\widehat{g_s}(\cdot)\}_{s=1,2,...,S}$,

$$y = \sum_{s=1}^{S} \widehat{g_s}(\mathcal{L}) x_s,$$

where x_s are the coefficients in the linear combination. For efficient representations in many applications, these coefficients should be sparse and they should capture the most important characteristics of the graph signals [7]. Sparsity however largely depends on the design of the dictionary, which in our signal model, depends on the choice of the polynomial coefficients.

2.2. Learning dictionaries for sparse representation

In many applications, someone has to deal with signals that live on different graphs but share some common spectral characteristics. Hence, we propose a dictionary learning algorithm that constructs graph atoms that are adapted to the representations of the signals on each graph independently, and simultaneously allow to capture the spectral similarities across the signals on the different graphs. In particular, we exploit the polynomial structure of the graph dictionaries, as defined in the previous subsection, and we capture the spectral components of the signals through the computation of kernels that are similar across all the graph topologies.

Given a set of training signals $Y_t = [y_{t1}, y_{t2}, ..., y_{tM_t}] \in \mathbb{R}^{N \times M_t}$, $t = \{1, 2, .., T\}$, living on the weighted graphs $\mathcal{G}_t, t = \{1, 2, ..., T\}$, our objective is to learn a common structure for the different graph dictionaries $\mathcal{D}_t \in \mathbb{R}^{N \times NS}$ that can efficiently represent all of the signals in $\mathcal{G}_t$ as linear combinations of only a few of its atoms. For each graph $\mathcal{G}_t$ and the corresponding training signals Y_t, we want to design a structured graph dictionary $\mathcal{D}_t = [\mathcal{D}_t^1, \mathcal{D}_t^2, ..., \mathcal{D}_t^S]$ that is a concatenation of a set of S subdictionaries of the form

$$\mathcal{D}_t^s = \widehat{g_s}(\mathcal{L}_t) = \chi_t \left(\sum_{k=0}^{K} \alpha_{sk} \Lambda_t^k \right) \chi_t^T = \sum_{k=0}^{K} \alpha_{sk} {\mathcal{L}_t}^k, \tag{4}$$

where $\mathcal{L}_t$ denotes the Laplacian of the graph t, and χ_t, Λ_t, are the corresponding eigenvectors and eigenvalues respectively. We thus impose a dictionary structure that is a polynomial function of the Laplacian and the coefficients of the polynomials, i.e., the generating kernel, capture the common information across the graphs in the spectral domain. The polynomial coefficients are learned jointly from the set of training signals on all graphs $\mathcal{G}_t$'s, in order to capture the similarity between graph signals independently of the actual graph topology. Therefore, the dictionary learning problem can be cast as the following optimization problem:

$$\begin{aligned} \underset{\alpha \in \mathbb{R}^{(K+1)S},\ X_t \in \mathbb{R}^{SN \times M_t}}{\operatorname{argmin}} & \left\{ \sum_{t=1}^{T} \frac{1}{M_t} ||Y_t - \mathcal{D}_t X_t||_F^2 + \mu \|\alpha\|_2^2 \right\} \\ \text{subject to} \quad & \|X_t^m\|_0 \leq T_0, \quad \forall m \in \{1, ..., M_t\}, \\ & \mathcal{D}_t^s = \sum_{k=0}^{K} \alpha_{sk} \mathcal{L}_t^k, \forall s \in \{1, 2, ..., S\}, \\ & 0 \preceq \mathcal{D}_t^s \preceq c, \forall s \in \{1, 2, ..., S\}, \end{aligned} \tag{5}$$

where X_t^m corresponds to column m of the coefficient matrix X_t, and T_0 is the maximum sparsity level of the coefficients of each training signal. Since $\mathcal{D}_t$ has the form (4), the optimization problem is equivalent to learning the parameters $\{\alpha_{sk}\}_{s=1,2,...,S;\ k=1,2,...,K}$ that characterize the set of generating kernels, $\{\widehat{g_s}(\cdot)\}_{s=1,2,...,S}$, and are the same across all the graphs. We denote these parameters in vector form in Eq. (5) with $\alpha = [\alpha_1; ...; \alpha_S]$, where α_s is a column vector with $(K+1)$ entries.

The constraint on the spectrum guarantees that the kernels are nonnegative and uniformly bounded by a given constant c. The value of the parameter c does not affect the frequency behavior nor the localization of the atoms. It simply scales the magnitude of the kernel

coefficients. Note that in the objective of the optimization problem (5), we penalize the norm of the polynomial coefficients α in order to (i) promote smoothness in the learned polynomial kernels, and (ii) improve the numerical stability of the learning algorithm. In practice, a small value of μ is enough to guarantee the stability of the solution while preserving large values in the polynomial coefficients.

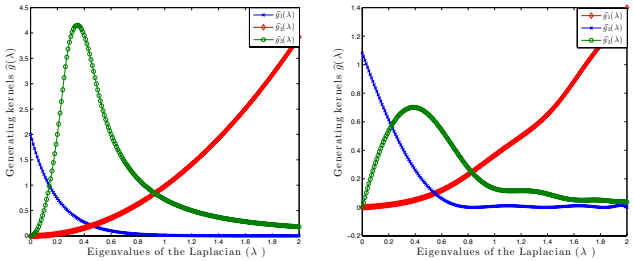
The optimization problem (5) is not jointly convex, but it can be approximately solved by alternating between the sparse coding and dictionary update steps. In the first step, we fix the dictionary and solve with respect to the sparse coding coefficients, using orthogonal matching pursuit (OMP) [11] on each training signal. Before applying OMP, we normalize the atoms of the dictionary so that they all have a unit norm. This step is essential for the OMP algorithm in order to treat all of the atoms equally. After computing the sparse codes, we renormalize the atoms of our dictionary to recover our initial polynomial structure [12, Chapter 3.1.4] and the sparse coding coefficients in such a way that the product of the dictionary and the sparse codes remains constant. In the second step, we fix the sparse coding coefficients and update the dictionary by solving the quadratic problem with respect to the parameters, α.

3. LEARNING OF COMMON GRAPH PROCESSES

In the following section, we quantify the performance of the proposed algorithm on learning dictionaries for synthetic data models that represent well-known processes. We build synthetic graph signals by choosing different graphs and computing the results of common diffusion processes on each of these graphs. We then learn dictionaries for these synthetic graph signals and show that our algorithm is able to recover the core components of the signals, which can be sparsely represented as combinations of few graph atoms.

We first construct three different types of graphs with 500 vertices, namely, a graph whose edges are determined based on Euclidean distances between vertices, and two graphs that follow the Forest Fire model [13] and the Barabási-Albert model [14], respectively. For the first graph, we generate the coordinates of the vertices uniformly at random in the unit square, and compute the edge weights between every pair of vertices using the Euclidean distances between them and a Gaussian radial basis function (RBF) $\exp\left(-d(i,j)^2/2\sigma^2\right)$, with the width parameter $\sigma = 0.04$. We then remove all the edges whose weights are smaller than 0.09. Next, we use the Forest Fire (FF) model with forward burning probability 0.1 to generate a random graph, and backward burning ratio 0.005. Finally, we use the Barabási-Albert (BA) model to generate a scale-free random graph. Then, we consider three data models that have been extensively used in the literature for applications such as classification [15], 3D shape analysis [16], and graph matching [17], for example. These models are the following:

1. *Heat diffusion kernel*: It is defined by choosing the kernel to be an exponential function of the eigenvalues of the Laplacian [15]:
$$\widehat{g}_\tau(\lambda_k) = e^{-\tau\lambda_k}. \tag{6}$$
Applying different powers τ of the heat diffusion operator to an initial signal x describes the flow of the heat over the graph when the rates of flows are proportional to the edge weights of the graph. Due to the exponential function of the kernel, this process mainly acts as a low frequency filtering, revealing information about the global behaviour of the signal.
2. *Wave kernel*: By selecting the kernel
$$\widehat{g}_\tau(\lambda_k) = e^{-\frac{(\tau-\log\lambda_k)^2}{2\sigma^2}}, \tag{7}$$
the process defined by (1), captures a different physical model that evaluates the average probability of a quantum particle with a certain energy distribution to be located at a particular vertex [18]. This model, contrary to the heat diffusion process, has a natural notion of scale defined by τ, since it is a function of the energy levels which are directly related to the scales. Thus, it is more appropriate for capturing localized information.
3. *Spectral graph wavelet kernel*: By selecting
$$\widehat{g}_\tau(\lambda_k) = g(\tau\lambda_k), \tag{8}$$
with $g(\lambda_k)$ properly chosen band pass filter, the process (1) becomes the spectral graph wavelet frame [4]. The different values of τ represent the different scales of the transform.



(a) Original kernels (b) Learned kernels

Fig. 1. Generating kernels $\{\widehat{g}_s(\cdot)\}_{s=1,2,\ldots,S}$ as a function of the eigenvalues.

With the above models, we then construct graph signals as follows. For each graph, we generate a synthetic dictionary consisting of $S = 3$ subdictionaries, i.e., 1500 atoms, of the form (1), generated by the following kernels: (i) a heat kernel, with $\tau = 5$, (ii), a wavelet kernel for a fixed scale of $\tau = 4.1$, which is set to be a cubic spline as defined in [4, Section 8.1, Eq. (65)], with $\alpha = \beta = 2$, $x_1 = 1$, $x_2 = 2$, and (iii) a wave kernel, with $\tau = 0.01$ and $\sigma = 1/\sqrt{2}$. The training signals are generated by linearly combining $T_0 = 4$ atoms from the corresponding dictionary, with randomly generated coefficients. We use these training signals to learn a graph dictionary built on polynomial functions of degree $K = 15$ of the Laplacian using our dictionary learning algorithm. In all our experiments, we use the *sdpt3* solver [19] in the *yalmip* optimization toolbox [20] to update the polynomial coefficients for fixed sparse codes in the learning algorithm. For the sparse coding step in the testing phase, we use OMP, where we first normalize the dictionary atoms to a unit norm. In all the experiments, we set $c = 10$, $\mu = 10^{-4}$, and the maximum number of iterations of the algorithm to 50.

The original and the learned kernels are illustrated in Fig. 1 as a function of the eigenvalues of the Laplacian, for $M = 400$. We observe that the learned kernels capture the spectral characteristics of the original ones that represent the three data models used to construct the training signals. It confirms that the polynomial atoms are able to correctly approximate the processes represented by the graph signals, despite the fact that these signals live on different graphs.

Then, we test the approximation performance of the learned dictionary on a set of 2000 testing signals for each graph, generated in the same way as the training signals. We further consider two different sizes for the training data: the training set in the first one consists of $M = 400$ signals per graph, while in the second one it consists

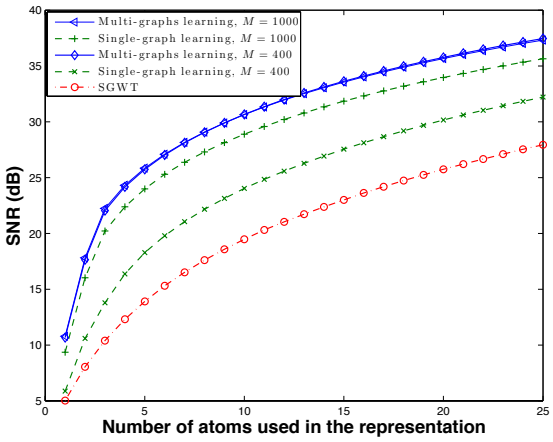


Fig. 2. Comparison of multi-graphs learning, with SGWT and single-graph learning in terms of SNR in the synthetic data.

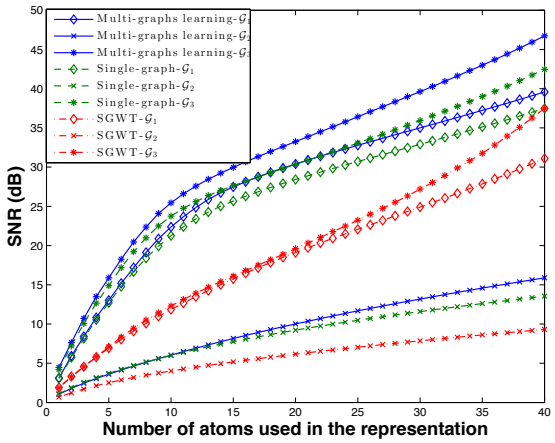


Fig. 3. Comparison of multi-graphs learning, with SGWT and single-graph learning in terms of SNR in the traffic delay dataset for three different counties.

of $M = 1000$ signals per graph. We compare the approximation performance of our algorithm to that obtained with (i) the spectral graph wavelet transform (SGWT) [4] and (ii) a polynomial graph dictionary, learned in each graph separately [7]. In Fig. 2, we plot the signal-to-approximation noise ratio (SNR) in dB, for different sparsity levels. The SNR for each sparsity level is the average over the three graphs. We observe that learning on multiple graphs can significantly improve the SNR with respect to both the SGWT and dictionaries learnt on different graphs independently. In particular, the results indicate that jointly learning a dictionary is more stable with respect to the size of the training set, and it can be more efficient than learning independently in each graph, especially when the training set is quite small ($M = 400$). In this case, the information obtained from the different graphs compensates for the lack of training signals in each graph separately and can be combined in order to learn the true dictionary.

4. SPARSE REPRESENTATION OF TRAFFIC DATA

In this subsection, we illustrate the performance of our algorithm in the representation of localized graph signals related to the traffic information in different counties in the state of California. In particular, we consider the daily bottlenecks in San Francisco, Alameda, and Santa Barbara counties between January 2007 and August 2014. A bottleneck could be any location where there is a persistent drop in speed, such as merges, large on-ramps, and incidents. The data are part of the Caltrans Performance Measurement System (PeMS) dataset that provides traffic information throughout all major metropolitan areas of California [21].[1] For each of the counties we design a graph whose nodes consist of $N = 75$, $N = 559$, and $N = 62$ detector stations respectively where bottlenecks were identified over the period under consideration. We have thus in total three different graphs, and each of them is designed by connecting stations when the distance between them is smaller than a threshold of $\theta = 0.04$. For two stations A, B, the distance d_{AB} is set to be the Euclidean distance of the GPS coordinates of the stations and the edge weights are computed using the exponential kernels such that $W_{AB} = e^{-d_{AB}}$. The signal on the graph is the duration in minutes of bottlenecks for each specific day. We remove the signal instances that no daily bottleneck was identified, and for computational issues, we normalize each signal to a unit norm. The final number of signals is 2766, 2772, and 894 for San Francisco ($\mathcal{G}_1$), Alameda ($\mathcal{G}_2$), and Santa Barbara ($\mathcal{G}_3$) respectively. For each graph, we use half of the signals for jointly training a polynomial graph dictionary and the rest for testing the performance of the learned dictionary. In our experiments, we fix the maximum degree of the polynomial to $K = 15$ and we learn a dictionary with $S = 3$.

In Fig. 3, we illustrate the sparse approximation performance of our dictionary representation by studying the reconstruction performance in SNR on the set of testing signals for different sparsity levels. The performance is compared to that obtained by learning separately a dictionary on each graph [7], and the one obtained by the sparse decomposition in the graph wavelet dictionary [4]. We observe that multi-graph learning improves significantly the performance in comparison to SGWT. Moreover, it outperforms the single-graph learning algorithm in all the graphs. In the latter case however, the gain is more evident when representing signals from the Santa Barbara graph $\mathcal{G}_3$. In this case, jointly learning a dictionary compensates for the relatively small number of available training signals for this graph in comparison to the other two graphs.

5. CONCLUSIONS

In this paper, we have proposed an algorithm for learning dictionaries to sparsely represent graph signals that share some common spectral characteristics, but live on different weighted graphs. These characteristics are reflected in a common dictionary structure that is constructed to be a polynomial function of the graph Laplacian matrix, and the coefficients of the polynomials are jointly learned from all graph signals on all graph instances. Experimental results show the effectiveness of our dictionary learning method in sparse representation of graph signals. Due to its polynomial construction, our dictionary is also independent from the actual graph topologies, which surely provides important benefits in terms of reduced complexity and increased flexibility in practical learning and processing methods for signal on graphs.

[1] The data are publicly available at http://pems.dot.ca.gov.

6. REFERENCES

[1] D. I Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[2] X. Zhu and M. Rabbat, "Approximating signals supported on graphs," in *Proc. IEEE Int. Conf. Acc., Speech, and Signal Process.*, Kyoto, Japan, Mar. 2012, pp. 3921–3924.

[3] R. R. Coifman and M. Maggioni, "Diffusion wavelets," *Appl. Comput. Harmon. Anal.*, vol. 21, pp. 53–94, Mar. 2006.

[4] D. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, Mar. 2010.

[5] S. K. Narang and A. Ortega, "Perfect reconstruction two-channel wavelet filter banks for graph structured data," *IEEE Trans. Signal Process.*, vol. 60, no. 6, pp. 2786–2799, June 2012.

[6] D. I Shuman, B. Ricaud, and P. Vandergheynst, "Vertex-frequency analysis on graphs," *arXiv preprint: http://arxiv.org/abs/1307.5708*, 2013.

[7] D. Thanou, D. I Shuman, and P. Frossard, "Learning parametric dictionaries for signals on graphs," *IEEE Trans. Signal Process.*, vol. 62, no. 15, pp. 3849–3862, Aug. 2014.

[8] X. Zhang, X. Dong, and P. Frossard, "Learning of structured graph dictionaries," in *Proc. IEEE Int. Conf. Acc., Speech, and Signal Process.*, Kyoto, Japan, Mar. 2012, pp. 3373 – 3376.

[9] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[10] F. Chung, *Spectral Graph Theory*, American Mathematical Society, 1997.

[11] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. Inform. Theory*, vol. 50, no. 10, pp. 2231–2242, Oct. 2004.

[12] M. Elad, *Sparse and Redundant Representations - From Theory to Applications in Signal and Image Processing*, Springer, 2010.

[13] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: Densification laws, shrinking diameters and possible explanations," in *Proc. of the 11th ACM SIGKDD Inter. Conf. on Knowledge Discovery in Data Mining*, Chicago, Illinois, USA, 2005, pp. 177–187.

[14] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, Oct 1999.

[15] R. Kondor and J. Lafferty, "Diffusion kernels on graphs and other discrete structures," in *Inter. Conf. of Machine Learn.*, 2002, pp. 315–322.

[16] W. H. Kim, M. K. Chung, and V. Singh, "Multi-resolution shape analysis via non-euclidean wavelets: Applications to mesh segmentation and surface alignment problems," in *IEEE Conf. on Comp. Vision and Pattern Recogn.*, 2013, pp. 2139–2146.

[17] N. Hu, R. M. Rustamov, and L. Guibas, "Stable and informative spectral signatures for graph matching," in *IEEE Conf. on Comp. Vision and Pattern Recogn.*, June 2014.

[18] M. Aubry, U. Schlickewei, and D. Cremers, "The wave kernel signature: A quantum mechanical approach to shape analysis," in *Proc. IEEE Int. Conf. Comp. Vision*, 2011.

[19] K.C. Toh, M.J. Todd, and R.H. Tutuncu, "SDPT3 — a MATLAB software package for semidefinite programming," in *Optimization Methods and Software*, 1999, vol. 11, pp. 545–581.

[20] J. Lofberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *Proc. CACSD Conf.*, Taipei, Taiwan, Sept. 2004.

[21] T. Choe, A. Skabardonis, and P. P. Varaiya, "Freeway performance measurement system (PeMS): an operational analysis tool," in *Annual Meeting of Transportation Research Board*, Washington, DC, USA, Jan. 2002.