The Hierarchical Graph Laplacian Eigen Transform (HGLET) and Its Relatives for Data Analysis on Graphs and Networks

Naoki Saito & Jeff Irion

Department of Mathematics
University of California, Davis

AMS Special Session on Applied Harmonic Analysis:
Large Data Sets, Signal Processing, and Inverse Problems

Joint Mathematics Meetings, Baltimore, MD

January 15, 2014

Motivations

Wavelets

- Have been quite successful on regular domains
- Have been extended to irregular domains ⇒ "2nd Generation Wavelets"

For example:

- Hammond, Vandergheynst, and Gribonval (2011): wavelets via spectral graph theory
- Coifman and Maggioni (2006): diffusion wavelets
    - Bremer *et al.* (2006): diffusion wavelet packets

Key difficulty: The notion of *frequency* is ill-defined on graphs $\Longrightarrow$ The Fourier transform is not properly defined on graphs

Common strategy: Develop wavelet-*like* multiscale transforms

Motivations

Wavelets

- Have been quite successful on regular domains
- Have been extended to irregular domains $\Rightarrow$ "2nd Generation Wavelets"

For example:

- Hammond, Vandergheynst, and Gribonval (2011): wavelets via spectral graph theory
- Coifman and Maggioni (2006): diffusion wavelets
    - Bremer *et al.* (2006): diffusion wavelet packets

Key difficulty: The notion of *frequency* is ill-defined on graphs $\Longrightarrow$ The Fourier transform is not properly defined on graphs

Common strategy: Develop wavelet-*like* multiscale transforms

Motivations

Wavelets

- Have been quite successful on regular domains
- Have been extended to irregular domains ⇒ "2nd Generation Wavelets"

For example:

- Hammond, Vandergheynst, and Gribonval (2011): wavelets via spectral graph theory
- Coifman and Maggioni (2006): diffusion wavelets
  - Bremer *et al.* (2006): diffusion wavelet packets

Key difficulty: The notion of *frequency* is ill-defined on graphs $\Longrightarrow$ The Fourier transform is not properly defined on graphs

Common strategy: Develop wavelet-*like* multiscale transforms

Motivations

Wavelets

- Have been quite successful on regular domains
- Have been extended to irregular domains ⇒ "2nd Generation Wavelets"

For example:

- Hammond, Vandergheynst, and Gribonval (2011): wavelets via spectral graph theory
- Coifman and Maggioni (2006): diffusion wavelets
  - Bremer *et al.* (2006): diffusion wavelet packets

Key difficulty: The notion of *frequency* is ill-defined on graphs $\Longrightarrow$ The Fourier transform is not properly defined on graphs

Common strategy: Develop wavelet-*like* multiscale transforms

Aims & Objectives

- Develop and implement multiscale transforms for data on graphs and networks; in particular, build *multiscale basis dictionaries* on graphs.
- Investigate their usefulness for a variety of applications including approximation, denoising, classification, and regression on graphs.
- In this talk, we will focus on how to construct such dictionaries on graphs and demonstrate their usefulness for data approximation on graphs.

Aims & Objectives

- Develop and implement multiscale transforms for data on graphs and networks; in particular, build *multiscale basis dictionaries* on graphs.
- Investigate their usefulness for a variety of applications including approximation, denoising, classification, and regression on graphs.
- In this talk, we will focus on how to construct such dictionaries on graphs and demonstrate their usefulness for data approximation on graphs.

Aims & Objectives

- Develop and implement multiscale transforms for data on graphs and networks; in particular, build *multiscale basis dictionaries* on graphs.
- Investigate their usefulness for a variety of applications including approximation, denoising, classification, and regression on graphs.
- In this talk, we will focus on how to construct such dictionaries on graphs and demonstrate their usefulness for data approximation on graphs.

Definitions and Notation

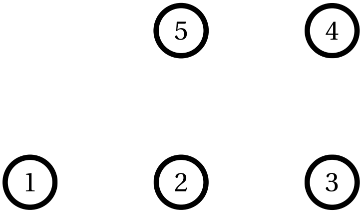
Let G be a graph.

- $V = V(G) = \{v_1, \ldots, v_N\}$ is the set of vertices.
- $E = E(G) = \{e_1, \ldots, e_{N'}\}$ is the set of edges, where $e_k = (v_i, v_j)$ represents an edge (or line segment) connecting between adjacent vertices v_i, v_j for some $1 \le i, j \le N$.
- $W = W(G) \in \mathbb{R}^{N \times N}$ is the weight matrix, where w_{ij} denotes the edge weight between vertices i and j.

Definitions and Notation

Let G be a graph.

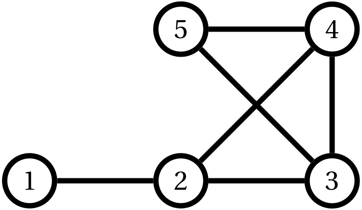
- $V = V(G) = \{v_1, \ldots, v_N\}$ is the set of vertices.
- $E = E(G) = \{e_1, \ldots, e_{N'}\}$ is the set of edges, where $e_k = (v_i, v_j)$ represents an edge (or line segment) connecting between adjacent vertices v_i, v_j for some $1 \le i, j \le N$.
- $W = W(G) \in \mathbb{R}^{N \times N}$ is the weight matrix, where w_{ij} denotes the edge weight between vertices i and j.



Definitions and Notation

Let G be a graph.

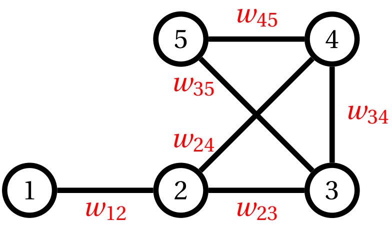
- $V = V(G) = \{v_1, \ldots, v_N\}$ is the set of vertices.
- $E = E(G) = \{e_1, \ldots, e_{N'}\}$ is the set of edges, where $e_k = (v_i, v_j)$ represents an edge (or line segment) connecting between adjacent vertices v_i, v_j for some $1 \le i, j \le N$.
- $W = W(G) \in \mathbb{R}^{N \times N}$ is the weight matrix, where w_{ij} denotes the edge weight between vertices i and j.



Definitions and Notation

Let G be a graph.

- $V = V(G) = \{v_1, \ldots, v_N\}$ is the set of vertices.
- $E = E(G) = \{e_1, \ldots, e_{N'}\}$ is the set of edges, where $e_k = (v_i, v_j)$ represents an edge (or line segment) connecting between adjacent vertices v_i, v_j for some $1 \le i, j \le N$.
- $W = W(G) \in \mathbb{R}^{N \times N}$ is the weight matrix, where w_{ij} denotes the edge weight between vertices i and j.



Definitions and Notation

Note that there are many ways to define w_{ij}.

For example, for *unweighted* graphs, we typically use

$$w_{ij} := \begin{cases} 1 & \text{if } v_i \sim v_j \text{ (i.e., } v_i \text{ and } v_j \text{ are adjacent);} \\ 0 & \text{otherwise.} \end{cases}$$

This is often referred to as the adjacency matrix and denoted by $A(G)$.

For *weighted* graphs, w_{ij} should reflect the similarity (or affinity) of information at v_i and v_j, e.g., if $v_i \sim v_j$, then

$$w_{ij} := 1/\mathrm{dist}(v_i, v_j) \quad \text{or} \quad \exp(-\mathrm{dist}(v_i, v_j)^2/\epsilon^2),$$

where $\mathrm{dist}(\cdot,\cdot)$ is a certain measure of dissimilarity and $\epsilon > 0$ is an appropriate scale parameter.

Definitions and Notation

Note that there are many ways to define w_{ij}.

For example, for *unweighted* graphs, we typically use

$$w_{ij} := \begin{cases} 1 & \text{if } v_i \sim v_j \text{ (i.e., } v_i \text{ and } v_j \text{ are adjacent);} \\ 0 & \text{otherwise.} \end{cases}$$

This is often referred to as the adjacency matrix and denoted by $A(G)$.

For *weighted* graphs, w_{ij} should reflect the similarity (or affinity) of information at v_i and v_j, e.g., if $v_i \sim v_j$, then

$$w_{ij} := 1/\mathrm{dist}(v_i, v_j) \quad \text{or} \quad \exp(-\mathrm{dist}(v_i, v_j)^2/\epsilon^2),$$

where $\mathrm{dist}(\cdot,\cdot)$ is a certain measure of dissimilarity and $\epsilon > 0$ is an appropriate scale parameter.

Our Assumptions

In this talk, we assume that the graph is

- **connected.** Otherwise, we would simply consider the components separately.
- **undirected.** Edges do not have direction, which means that $w_{ij} = w_{ji}$ and thus W is *symmetric*.

The graph may be weighted or unweighted.

Our Assumptions

In this talk, we assume that the graph is

- **connected.** Otherwise, we would simply consider the components separately.
- **undirected.** Edges do not have direction, which means that $w_{ij} = w_{ji}$ and thus W is *symmetric*.

The graph may be weighted or unweighted.

Our Assumptions

In this talk, we assume that the graph is

- **connected.** Otherwise, we would simply consider the components separately.
- **undirected.** Edges do not have direction, which means that $w_{ij} = w_{ji}$ and thus W is *symmetric*.

The graph may be weighted or unweighted.

Our Assumptions

In this talk, we assume that the graph is

- **connected.** Otherwise, we would simply consider the components separately.
- **undirected.** Edges do not have direction, which means that $w_{ij} = w_{ji}$ and thus W is *symmetric*.

The graph may be weighted or unweighted.

Graph Laplacians

$$\begin{cases} W(G) = (w_{ij}) & \text{the weight matrix} \\ D(G) := \mathrm{diag}(d_{v_1}, \ldots, d_{v_n}) & \text{the degree matrix, where } d_{v_i} := \sum_{j=1}^{N} w_{ij}. \\ L(G) := D(G) - W(G) & \text{the Laplacian matrix} \end{cases}$$

We have:

- sorted eigenvalues $0 = \lambda_0 < \lambda_1 \le \cdots \le \lambda_{N-1}$
- associated eigenvectors $\boldsymbol{\phi}_0,\ \boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{N-1}$

The eigenvectors form a basis for $\mathbb{R}^N$. In particular:

- since L is symmetric, the eigenvectors form an orthonormal basis
- $\boldsymbol{\phi}_0 = \mathbf{1}/\sqrt{N}$

Graph Laplacians

$$\begin{cases} W(G) = (w_{ij}) & \text{the weight matrix} \\ D(G) := \operatorname{diag}(d_{v_1}, \ldots, d_{v_n}) & \text{the degree matrix, where } d_{v_i} := \sum_{j=1}^{N} w_{ij}. \\ L(G) := D(G) - W(G) & \text{the Laplacian matrix} \end{cases}$$

We have:

- sorted eigenvalues $0 = \lambda_0 < \lambda_1 \leq \cdots \leq \lambda_{N-1}$
- associated eigenvectors $\boldsymbol{\phi}_0,\ \boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{N-1}$

The eigenvectors form a basis for $\mathbb{R}^N$. In particular:

- since L is symmetric, the eigenvectors form an orthonormal basis
- $\boldsymbol{\phi}_0 = \mathbf{1}/\sqrt{N}$

Graph Laplacians

$$\begin{cases} W(G) = (w_{ij}) & \text{the weight matrix} \\ D(G) := \mathrm{diag}(d_{v_1}, \ldots, d_{v_n}) & \text{the degree matrix, where } d_{v_i} := \sum_{j=1}^{N} w_{ij}. \\ L(G) := D(G) - W(G) & \text{the Laplacian matrix} \end{cases}$$

We have:

- sorted eigenvalues $0 = \lambda_0 < \lambda_1 \leq \cdots \leq \lambda_{N-1}$
- associated eigenvectors $\boldsymbol{\phi}_0,\ \boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{N-1}$

The eigenvectors form a basis for $\mathbb{R}^N$. In particular:

- since L is symmetric, the eigenvectors form an orthonormal basis
- $\boldsymbol{\phi}_0 = \mathbf{1}/\sqrt{N}$

Why Graph Laplacians?

- Let $f \in \mathbb{R}^N$. Then

$$Lf(v_i) = d_{v_i} f(v_i) - \sum_{j \neq i} w_{ij} f(v_j).$$

  This is a generalization of *the finite difference approximation to the Laplace operator*.

- After all, *sines* (*cosines*) are the eigenfunctions of the Laplacian on the rectangular domain with Dirichlet (Neumann) boundary conditions.
- Hence, the expansion of data measured at the vertices using the eigenvectors of a graph Laplacian can be viewed as *Fourier (or spectral) analysis of the data on that graph*.

Why Graph Laplacians?

- Let $f \in \mathbb{R}^N$. Then

$$Lf(v_i) = d_{v_i} f(v_i) - \sum_{j \neq i} w_{ij} f(v_j).$$

  This is a generalization of *the finite difference approximation to the Laplace operator*.

- After all, *sines* (*cosines*) are the eigenfunctions of the Laplacian on the rectangular domain with Dirichlet (Neumann) boundary conditions.
- Hence, the expansion of data measured at the vertices using the eigenvectors of a graph Laplacian can be viewed as *Fourier (or spectral) analysis of the data on that graph*.

Why Graph Laplacians?

- Let $f \in \mathbb{R}^N$. Then

$$Lf(v_i) = d_{v_i} f(v_i) - \sum_{j \neq i} w_{ij} f(v_j).$$

  This is a generalization of *the finite difference approximation to the Laplace operator*.
- After all, *sines* (*cosines*) are the eigenfunctions of the Laplacian on the rectangular domain with Dirichlet (Neumann) boundary conditions.
- Hence, the expansion of data measured at the vertices using the eigenvectors of a graph Laplacian can be viewed as *Fourier (or spectral) analysis of the data on that graph*.

A Simple Yet Important Example: A Path Graph

$$\underbrace{\begin{bmatrix} 1 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{bmatrix}}_{L(G)} = \underbrace{\begin{bmatrix} 1 & & & & & \\ & 2 & & & & \\ & & 2 & & & \\ & & & \ddots & & \\ & & & & 2 & \\ & & & & & 1 \end{bmatrix}}_{D(G)} - \underbrace{\begin{bmatrix} 0 & 1 & & & & \\ 1 & 0 & 1 & & & \\ & 1 & 0 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 0 & 1 \\ & & & & 1 & 0 \end{bmatrix}}_{A(G)}$$

The eigenvectors of this matrix are exactly the *DCT Type II* basis vectors used for the JPEG image compression standard! (See e.g., Strang, SIAM Review, 1999).

- $\lambda_k = 2 - 2\cos(\pi k/N) = 4\sin^2(\pi k/2N)$, $k = 0, 1, \ldots, N-1$.
- $\phi_k(\ell) = \sqrt{2/N}\cos\left(\pi k(\ell + \frac{1}{2})/N\right)$, $k, \ell = 0, 1, \ldots, N-1$.
- λ (eigenvalue) is a monotonic function w.r.t. k (frequency). However, for general graphs, λ does not have a simple relationship with k.

A Simple Yet Important Example: A Path Graph

$$\underbrace{\begin{bmatrix} 1 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{bmatrix}}_{L(G)} = \underbrace{\begin{bmatrix} 1 & & & & & \\ & 2 & & & & \\ & & 2 & & & \\ & & & \ddots & & \\ & & & & 2 & \\ & & & & & 1 \end{bmatrix}}_{D(G)} - \underbrace{\begin{bmatrix} 0 & 1 & & & & \\ 1 & 0 & 1 & & & \\ & 1 & 0 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 0 & 1 \\ & & & & 1 & 0 \end{bmatrix}}_{A(G)}$$

The eigenvectors of this matrix are exactly the *DCT Type II* basis vectors used for the JPEG image compression standard! (See e.g., Strang, SIAM Review, 1999).

- $\lambda_k = 2 - 2\cos(\pi k/N) = 4\sin^2(\pi k/2N), \; k = 0, 1, \ldots, N-1.$
- $\boldsymbol{\phi}_k(\ell) = \sqrt{2/N}\cos\left(\pi k(\ell + \frac{1}{2})/N\right), \; k, \ell = 0, 1, \ldots, N-1.$
- λ (eigenvalue) is a monotonic function w.r.t. k (frequency). However, for general graphs, λ does not have a simple relationship with k.

Goal: split the vertices V into two "good" subsets, X and X^c

Plan: use the signs of the entries in $\boldsymbol{\phi}_1$, which is known as the Fiedler vector

Why? Using $\boldsymbol{\phi}_1$ to generate X and X^c yields an approximate minimizer of the RatioCut function[1,2]:

$$\mathrm{RatioCut}(X, X^c) := \frac{\mathrm{cut}(X, X^c)}{|X|} + \frac{\mathrm{cut}(X, X^c)}{|X^c|},$$

where

$$\mathrm{cut}(X, X^c) := \sum_{\substack{v_i \in X \\ v_j \in X^c}} W_{ij}$$

[1] L. Hagen and A. B. Kahng: "New spectral methods for ratio cut partitioning and clustering," *IEEE Trans. Comput.-Aided Des.*, vol. 11, no. 9, pp. 1074-1085, 1992.

[2] We could also use the signs of $\boldsymbol{\phi}_1$ for L_{rw} (equivalently, L_{sym}), which yield an approximate minimizer of the Normalized Cut function.

Goal: split the vertices V into two "good" subsets, X and X^c

Plan: use the signs of the entries in $\boldsymbol{\phi}_1$, which is known as the Fiedler vector

Why? Using $\boldsymbol{\phi}_1$ to generate X and X^c yields an approximate minimizer of the RatioCut function[1,2]:

$$\mathrm{RatioCut}(X, X^c) := \frac{\mathrm{cut}(X, X^c)}{|X|} + \frac{\mathrm{cut}(X, X^c)}{|X^c|},$$

where

$$\mathrm{cut}(X, X^c) := \sum_{\substack{v_i \in X \\ v_j \in X^c}} W_{ij}$$

[1] L. Hagen and A. B. Kahng: "New spectral methods for ratio cut partitioning and clustering," *IEEE Trans. Comput.-Aided Des.*, vol. 11, no. 9, pp. 1074-1085, 1992.

[2] We could also use the signs of $\boldsymbol{\phi}_1$ for L_{rw} (equivalently, L_{sym}), which yield an approximate minimizer of the Normalized Cut function.

Goal: split the vertices V into two "good" subsets, X and X^c

Plan: use the signs of the entries in $\boldsymbol{\phi}_1$, which is known as the Fiedler vector

Why? Using $\boldsymbol{\phi}_1$ to generate X and X^c yields an approximate minimizer of the RatioCut function[1,2]:

$$\mathrm{RatioCut}(X, X^c) := \frac{\mathrm{cut}(X, X^c)}{|X|} + \frac{\mathrm{cut}(X, X^c)}{|X^c|},$$

where

$$\mathrm{cut}(X, X^c) := \sum_{\substack{v_i \in X \\ v_j \in X^c}} W_{ij}$$

[1] L. Hagen and A. B. Kahng: "New spectral methods for ratio cut partitioning and clustering," *IEEE Trans. Comput.-Aided Des.*, vol. 11, no. 9, pp. 1074-1085, 1992.

[2] We could also use the signs of $\boldsymbol{\phi}_1$ for L_{rw} (equivalently, L_{sym}), which yield an approximate minimizer of the Normalized Cut function.

Example of Graph Partitioning

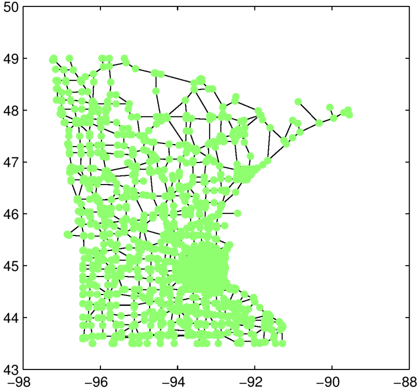


Figure: The MN road network

Example of Graph Partitioning

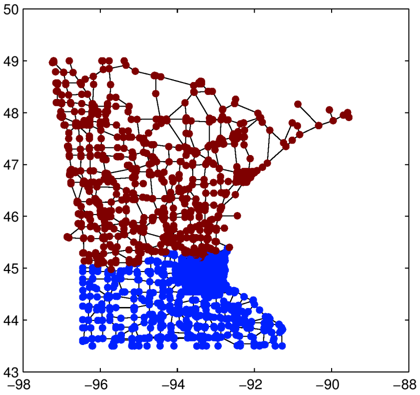


Figure: The MN road network partitioned via the Fiedler vector of L

Our transforms involve 2 main steps:

1. Recursively partition the graph

⇕ These steps can be performed concurrently, or we can fully partition the graph and then generate a set of bases

2. Using the regions on each level of the graph partitioning, generate a set of orthonormal bases for the graph

Our transforms involve 2 main steps:

1. Recursively partition the graph

$\Updownarrow$ These steps can be performed concurrently, or we can fully partition the graph and then generate a set of bases

2. Using the regions on each level of the graph partitioning, generate a set of orthonormal bases for the graph

Hierarchical Graph Laplacian Eigen Transform (HGLET)

Now we present a novel transform that can be viewed as a generalization of the *block Discrete Cosine Transform*. We refer to this transform as the *Hierarchical Graph Laplacian Eigen Transform (HGLET)*.

The algorithm proceeds as follows...

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}^j_{k,l}$ with $j = 0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}^j_{k,1}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N^0_0-1} \end{array} \right]$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}^{j}_{k,l}$ with $j = 0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}^{j}_{k,1}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}^{0}_{0,0} & \boldsymbol{\phi}^{0}_{0,1} & \boldsymbol{\phi}^{0}_{0,2} & \cdots & \boldsymbol{\phi}^{0}_{0,N^0_0-1} \end{array} \right]$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}^j_{k,l}$ with $j = 0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}^j_{k,1}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N^0_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N^1_1-1} \end{bmatrix}$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}^{j}_{k,l}$ with $j = 0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}^{j}_{k,1}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\begin{bmatrix} \boldsymbol{\phi}^{0}_{0,0} & \boldsymbol{\phi}^{0}_{0,1} & \boldsymbol{\phi}^{0}_{0,2} & \cdots & \boldsymbol{\phi}^{0}_{0,N^0_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^{1}_{0,0} & \boldsymbol{\phi}^{1}_{0,1} & \boldsymbol{\phi}^{1}_{0,2} & \cdots & \boldsymbol{\phi}^{1}_{0,N^1_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^{1}_{1,0} & \boldsymbol{\phi}^{1}_{1,1} & \boldsymbol{\phi}^{1}_{1,2} & \cdots & \boldsymbol{\phi}^{1}_{1,N^1_1-1} \end{bmatrix}$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}^{j}_{k,l}$ with $j = 0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}^{j}_{k,1}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\begin{bmatrix} \boldsymbol{\phi}^{0}_{0,0} & \boldsymbol{\phi}^{0}_{0,1} & \boldsymbol{\phi}^{0}_{0,2} & \cdots & \boldsymbol{\phi}^{0}_{0,N^0_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^{1}_{0,0} & \boldsymbol{\phi}^{1}_{0,1} & \boldsymbol{\phi}^{1}_{0,2} & \cdots & \boldsymbol{\phi}^{1}_{0,N^1_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^{1}_{1,0} & \boldsymbol{\phi}^{1}_{1,1} & \boldsymbol{\phi}^{1}_{1,2} & \cdots & \boldsymbol{\phi}^{1}_{1,N^1_1-1} \end{bmatrix}$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}^{j}_{k,l}$ with $j = 0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}^{j}_{k,1}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N^0_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N^1_1-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{0,0}\, \boldsymbol{\phi}^2_{0,1} \cdots \boldsymbol{\phi}^2_{0,N^2_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{1,0}\, \boldsymbol{\phi}^2_{1,1} \cdots \boldsymbol{\phi}^2_{1,N^2_1-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{2,0}\, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N^2_2-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{3,0}\, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N^2_3-1} \end{bmatrix}$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}^{j}_{k,l}$ with $j = 0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}^{j}_{k,1}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\ \boldsymbol{\phi}^{0}_{0,0} \quad \boldsymbol{\phi}^{0}_{0,1} \quad \boldsymbol{\phi}^{0}_{0,2} \quad \cdots \quad \boldsymbol{\phi}^{0}_{0,N^0_0-1}\ \right]$$

$$\left[\ \boldsymbol{\phi}^{1}_{0,0} \ \ \boldsymbol{\phi}^{1}_{0,1} \ \ \boldsymbol{\phi}^{1}_{0,2} \ \ \cdots \ \ \boldsymbol{\phi}^{1}_{0,N^1_0-1}\ \right] \quad \left[\ \boldsymbol{\phi}^{1}_{1,0} \ \ \boldsymbol{\phi}^{1}_{1,1} \ \ \boldsymbol{\phi}^{1}_{1,2} \ \ \cdots \ \ \boldsymbol{\phi}^{1}_{1,N^1_1-1}\ \right]$$

$$\left[\boldsymbol{\phi}^{2}_{0,0}\,\boldsymbol{\phi}^{2}_{0,1}\cdots\boldsymbol{\phi}^{2}_{0,N^2_0-1}\right] \left[\boldsymbol{\phi}^{2}_{1,0}\,\boldsymbol{\phi}^{2}_{1,1}\cdots\boldsymbol{\phi}^{2}_{1,N^2_1-1}\right] \left[\boldsymbol{\phi}^{2}_{2,0}\,\boldsymbol{\phi}^{2}_{2,1}\cdots\boldsymbol{\phi}^{2}_{2,N^2_2-1}\right] \left[\boldsymbol{\phi}^{2}_{3,0}\,\boldsymbol{\phi}^{2}_{3,1}\cdots\boldsymbol{\phi}^{2}_{3,N^2_3-1}\right]$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}^j_{k,l}$ with $j = 0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}^j_{k,1}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N^0_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N^1_1-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{0,0}\, \boldsymbol{\phi}^2_{0,1} \cdots \boldsymbol{\phi}^2_{0,N^2_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{1,0}\, \boldsymbol{\phi}^2_{1,1} \cdots \boldsymbol{\phi}^2_{1,N^2_1-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{2,0}\, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N^2_2-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{3,0}\, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N^2_3-1} \end{bmatrix}$$

$$\vdots$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}^{j}_{k,l}$ with $j = 0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}^{j}_{k,1}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N^0_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N^1_1-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{0,0}\, \boldsymbol{\phi}^2_{0,1} \cdots \boldsymbol{\phi}^2_{0,N^2_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{1,0}\, \boldsymbol{\phi}^2_{1,1} \cdots \boldsymbol{\phi}^2_{1,N^2_1-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{2,0}\, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N^2_2-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{3,0}\, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N^2_3-1} \end{bmatrix}$$

$$\vdots$$

Remarks

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, . . .
    - A union of bases on disjoint subsets is obviously orthonormal

Remarks

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal

Remarks

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand $\Rightarrow$ best-basis algorithm, local discriminant basis algorithm, . . .
  - A union of bases on disjoint subsets is obviously orthonormal

Remarks

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, . . .
    - A union of bases on disjoint subsets is obviously orthonormal

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N^0_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N^1_1-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{0,0} & \cdots & \boldsymbol{\phi}^2_{0,N^2_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{1,0} & \cdots & \boldsymbol{\phi}^2_{1,N^2_1-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{2,0} & \cdots & \boldsymbol{\phi}^2_{2,N^2_2-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} & \cdots & \boldsymbol{\phi}^2_{3,N^2_3-1} \end{bmatrix}$$

Remarks

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, . . .
    - A union of bases on disjoint subsets is obviously orthonormal

$$\left[\ \boldsymbol{\phi}^0_{0,0} \quad \boldsymbol{\phi}^0_{0,1} \quad \boldsymbol{\phi}^0_{0,2} \quad \cdots \quad \boldsymbol{\phi}^0_{0,N^0_0-1}\ \right]$$

$$\left[\boldsymbol{\phi}^1_{0,0} \quad \boldsymbol{\phi}^1_{0,1} \quad \boldsymbol{\phi}^1_{0,2} \quad \cdots \quad \boldsymbol{\phi}^1_{0,N^1_0-1}\right] \left[\boldsymbol{\phi}^1_{1,0} \quad \boldsymbol{\phi}^1_{1,1} \quad \boldsymbol{\phi}^1_{1,2} \quad \cdots \quad \boldsymbol{\phi}^1_{1,N^1_1-1}\right]$$

$$\left[\boldsymbol{\phi}^2_{0,0} \ \cdots \ \boldsymbol{\phi}^2_{0,N^2_0-1}\right] \left[\boldsymbol{\phi}^2_{1,0} \ \cdots \ \boldsymbol{\phi}^2_{1,N^2_1-1}\right] \left[\boldsymbol{\phi}^2_{2,0} \ \cdots \ \boldsymbol{\phi}^2_{2,N^2_2-1}\right] \left[\boldsymbol{\phi}^2_{3,0} \ \cdots \ \boldsymbol{\phi}^2_{3,N^2_3-1}\right]$$

Remarks

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N^0_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N^1_1-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{0,0} & \cdots & \boldsymbol{\phi}^2_{0,N^2_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{1,0} & \cdots & \boldsymbol{\phi}^2_{1,N^2_1-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{2,0} & \cdots & \boldsymbol{\phi}^2_{2,N^2_2-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} & \cdots & \boldsymbol{\phi}^2_{3,N^2_3-1} \end{bmatrix}$$

Remarks

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal

$$\left[\; \boldsymbol{\phi}^0_{0,0} \quad \boldsymbol{\phi}^0_{0,1} \quad \boldsymbol{\phi}^0_{0,2} \quad \cdots \quad \boldsymbol{\phi}^0_{0,N^0_0-1} \;\right]$$

$$\left[\boldsymbol{\phi}^1_{0,0} \quad \boldsymbol{\phi}^1_{0,1} \quad \boldsymbol{\phi}^1_{0,2} \quad \cdots \quad \boldsymbol{\phi}^1_{0,N^1_0-1}\right] \left[\boldsymbol{\phi}^1_{1,0} \quad \boldsymbol{\phi}^1_{1,1} \quad \boldsymbol{\phi}^1_{1,2} \quad \cdots \quad \boldsymbol{\phi}^1_{1,N^1_1-1}\right]$$

$$\left[\boldsymbol{\phi}^2_{0,0} \;\cdots\; \boldsymbol{\phi}^2_{0,N^2_0-1}\right] \left[\boldsymbol{\phi}^2_{1,0} \;\cdots\; \boldsymbol{\phi}^2_{1,N^2_1-1}\right] \left[\boldsymbol{\phi}^2_{2,0} \;\cdots\; \boldsymbol{\phi}^2_{2,N^2_2-1}\right] \left[\boldsymbol{\phi}^2_{3,0} \;\cdots\; \boldsymbol{\phi}^2_{3,N^2_3-1}\right]$$

Remarks

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, . . .
  - A union of bases on disjoint subsets is obviously orthonormal

$$\left[\ \boldsymbol{\phi}^0_{0,0} \quad \boldsymbol{\phi}^0_{0,1} \quad \boldsymbol{\phi}^0_{0,2} \quad \cdots \quad \boldsymbol{\phi}^0_{0,N^0_0-1}\ \right]$$

$$\color{red}{\left[\boldsymbol{\phi}^1_{0,0} \quad \boldsymbol{\phi}^1_{0,1} \quad \boldsymbol{\phi}^1_{0,2} \quad \cdots \quad \boldsymbol{\phi}^1_{0,N^1_0-1}\right]} \left[\boldsymbol{\phi}^1_{1,0} \quad \boldsymbol{\phi}^1_{1,1} \quad \boldsymbol{\phi}^1_{1,2} \quad \cdots \quad \boldsymbol{\phi}^1_{1,N^1_1-1}\right]$$

$$\left[\boldsymbol{\phi}^2_{0,0} \ \cdots \ \boldsymbol{\phi}^2_{0,N^2_0-1}\right] \left[\boldsymbol{\phi}^2_{1,0} \ \cdots \ \boldsymbol{\phi}^2_{1,N^2_1-1}\right] \color{red}{\left[\boldsymbol{\phi}^2_{2,0} \ \cdots \ \boldsymbol{\phi}^2_{2,N^2_2-1}\right] \left[\boldsymbol{\phi}^2_{3,0} \ \cdots \ \boldsymbol{\phi}^2_{3,N^2_3-1}\right]}$$

Remarks

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N^0_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{bmatrix} \color{red}{\begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N^1_1-1} \end{bmatrix}}$$

$$\color{red}{\begin{bmatrix} \boldsymbol{\phi}^2_{0,0} & \cdots & \boldsymbol{\phi}^2_{0,N^2_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{1,0} & \cdots & \boldsymbol{\phi}^2_{1,N^2_1-1} \end{bmatrix}} \begin{bmatrix} \boldsymbol{\phi}^2_{2,0} & \cdots & \boldsymbol{\phi}^2_{2,N^2_2-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} & \cdots & \boldsymbol{\phi}^2_{3,N^2_3-1} \end{bmatrix}$$

Related Work

The following work also proposed the similar strategy to construct a multiscale basis dictionary, i.e., *local cosine dictionary on a graph*:

1. A. D. Szlam, M. Maggioni, R. R. Coifman, and J. C. Bremer, Jr., "Diffusion-driven multiscale analysis on manifolds and graphs: top-down and bottom-up constructions," in *Wavelets XI* (M. Papadakis et al. eds.), *Proc. SPIE 5914*, Paper # 59141D, 2005.

Related Work

The following work also proposed the similar strategy to construct a multiscale basis dictionary, i.e., *local cosine dictionary on a graph*:

1. A. D. Szlam, M. Maggioni, R. R. Coifman, and J. C. Bremer, Jr., "Diffusion-driven multiscale analysis on manifolds and graphs: top-down and bottom-up constructions," in *Wavelets XI* (M. Papadakis et al. eds.), *Proc. SPIE 5914*, Paper # 59141D, 2005.

However, in our opinion, the generalization of the folding/unfolding operations (originally used in the construction of the local cosine transforms on a regular domain) to the graph setting may be harmful. We believe that such operations are not necessary for the most tasks in practice. If one needs smoother and overlapping basis vectors, then a better partitioning scheme other than the folding/unfolding operations is called for.

Computational Complexity: HGLET

	Computational Complexity	Run Time for MN[1]
HGLET (redundant)	$O(N^3)$	67 sec

[1] Computations performed on a personal laptop (4.00 GB RAM, 2.26 GHz), $N = 2640$ and `nnz(W)` $= 6604$.

Generalized Haar-Walsh Transform (GHWT)

HGLET is a generalization of the block DCT, and it generates basis vectors that are *smooth on their support*.

The Generalized Haar-Walsh Transform (GHWT) is a generalization of the classical Haar and Walsh-Hadamard Transforms, and it generates basis vectors that are *piecewise-constant on their support*.

The algorithm proceeds as follows...

Generalized Haar-Walsh Transform (GHWT)

HGLET is a generalization of the block DCT, and it generates basis vectors that are *smooth on their support*.

The Generalized Haar-Walsh Transform (GHWT) is a generalization of the classical Haar and Walsh-Hadamard Transforms, and it generates basis vectors that are *piecewise-constant on their support*.

The algorithm proceeds as follows...

1. Generate a full recursive partitioning of the graph ⇒ Fiedler vectors
2. Generate an orthonormal basis for level $j_{\max}$ (the finest level) ⇒ ***scaling vectors*** on the single-node regions
   - As with HGLET, the notation is $\boldsymbol{\psi}^j_{k,l}$
3. Using the basis for level $j_{\max}$, generate an orthonormal basis for level $j_{\max}-1$ ⇒ ***scaling*** and ***Haar-like*** vectors
4. Repeat... Using the basis for level j, generate an orthonormal basis for level $j-1$ ⇒ ***scaling***, ***Haar-like***, and ***Walsh-like*** vectors
5. Select an orthonormal basis from this collection of orthonormal bases

1. Generate a full recursive partitioning of the graph ⇒ Fiedler vectors
2. Generate an orthonormal basis for level $j_{\max}$ (the finest level) ⇒ ***scaling vectors*** on the single-node regions
   - As with HGLET, the notation is $\boldsymbol{\psi}^{j}_{k,l}$
3. Using the basis for level $j_{\max}$, generate an orthonormal basis for level $j_{\max}-1$ ⇒ ***scaling*** and ***Haar-like*** vectors
4. Repeat... Using the basis for level j, generate an orthonormal basis for level $j-1$ ⇒ ***scaling***, ***Haar-like***, and ***Walsh-like*** vectors
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\ \boldsymbol{\psi}^{j_{\max}}_{0,0}\ \right] \quad \left[\ \boldsymbol{\psi}^{j_{\max}}_{1,0}\ \right] \quad \left[\ \boldsymbol{\psi}^{j_{\max}}_{2,0}\ \right] \quad \left[\ \boldsymbol{\psi}^{j_{\max}}_{3,0}\ \right] \cdots \left[\ \boldsymbol{\psi}^{j_{\max}}_{K^{j_{\max}}-2,0}\ \right] \quad \left[\ \boldsymbol{\psi}^{j_{\max}}_{K^{j_{\max}}-1,0}\ \right]$$

1. Generate a full recursive partitioning of the graph ⇒ Fiedler vectors
2. Generate an orthonormal basis for level $j_{\max}$ (the finest level) ⇒ ***scaling vectors*** on the single-node regions
   - As with HGLET, the notation is $\boldsymbol{\psi}^{j}_{k,l}$
3. Using the basis for level $j_{\max}$, generate an orthonormal basis for level $j_{\max}-1$ ⇒ ***scaling*** and ***Haar-like*** vectors
4. Repeat... Using the basis for level j, generate an orthonormal basis for level $j-1$ ⇒ ***scaling***, ***Haar-like***, and ***Walsh-like*** vectors
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\ \boldsymbol{\psi}^{j_{\max}-1}_{0,0}\quad \boldsymbol{\psi}^{j_{\max}-1}_{0,1}\ \right]\ \left[\ \boldsymbol{\psi}^{j_{\max}-1}_{1,0}\quad \boldsymbol{\psi}^{j_{\max}-1}_{1,1}\ \right]\ \cdots\ \left[\ \boldsymbol{\psi}^{j_{\max}-1}_{K^{j_{\max}-1}-1,0}\quad \boldsymbol{\psi}^{j_{\max}-1}_{K^{j_{\max}-1}-1,1}\ \right]$$

$$\left[\ \boldsymbol{\psi}^{j_{\max}}_{0,0}\ \right]\ \left[\ \boldsymbol{\psi}^{j_{\max}}_{1,0}\ \right]\ \left[\ \boldsymbol{\psi}^{j_{\max}}_{2,0}\ \right]\ \left[\ \boldsymbol{\psi}^{j_{\max}}_{3,0}\ \right]\ \cdots\ \left[\ \boldsymbol{\psi}^{j_{\max}}_{K^{j_{\max}}-2,0}\ \right]\ \left[\ \boldsymbol{\psi}^{j_{\max}}_{K^{j_{\max}}-1,0}\ \right]$$

1. Generate a full recursive partitioning of the graph ⇒ Fiedler vectors
2. Generate an orthonormal basis for level $j_{\max}$ (the finest level) ⇒ ***scaling vectors*** on the single-node regions
   - As with HGLET, the notation is $\boldsymbol{\psi}_{k,l}^{j}$
3. Using the basis for level $j_{\max}$, generate an orthonormal basis for level $j_{\max}-1$ ⇒ ***scaling*** and ***Haar-like*** vectors
4. Repeat... Using the basis for level j, generate an orthonormal basis for level $j-1$ ⇒ ***scaling***, ***Haar-like***, and ***Walsh-like*** vectors
5. Select an orthonormal basis from this collection of orthonormal bases

$$\begin{bmatrix} \boldsymbol{\psi}_{0,0}^{0} & \boldsymbol{\psi}_{0,1}^{0} & \boldsymbol{\psi}_{0,2}^{0} & \boldsymbol{\psi}_{0,3}^{0} & \cdots & \boldsymbol{\psi}_{0,N-2}^{0} & \boldsymbol{\psi}_{0,N-1}^{0} \end{bmatrix}$$

$$\vdots$$

$$\begin{bmatrix} \boldsymbol{\psi}_{0,0}^{j_{\max}-1} & \boldsymbol{\psi}_{0,1}^{j_{\max}-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_{1,0}^{j_{\max}-1} & \boldsymbol{\psi}_{1,1}^{j_{\max}-1} \end{bmatrix} \cdots \begin{bmatrix} \boldsymbol{\psi}_{K^{j_{\max}-1}-1,0}^{j_{\max}-1} & \boldsymbol{\psi}_{K^{j_{\max}-1}-1,1}^{j_{\max}-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\psi}_{0,0}^{j_{\max}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_{1,0}^{j_{\max}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_{2,0}^{j_{\max}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_{3,0}^{j_{\max}} \end{bmatrix} \cdots \begin{bmatrix} \boldsymbol{\psi}_{K^{j_{\max}}-2,0}^{j_{\max}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_{K^{j_{\max}}-1,0}^{j_{\max}} \end{bmatrix}$$

1. Generate a full recursive partitioning of the graph ⇒ Fiedler vectors
2. Generate an orthonormal basis for level $j_{\max}$ (the finest level) ⇒ ***scaling vectors*** on the single-node regions
   - As with HGLET, the notation is $\boldsymbol{\psi}^{j}_{k,l}$
3. Using the basis for level $j_{\max}$, generate an orthonormal basis for level $j_{\max}-1$ ⇒ ***scaling*** and ***Haar-like*** vectors
4. Repeat... Using the basis for level j, generate an orthonormal basis for level $j-1$ ⇒ ***scaling***, ***Haar-like***, and ***Walsh-like*** vectors
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\ \boldsymbol{\psi}^0_{0,0} \quad \boldsymbol{\psi}^0_{0,1} \quad \boldsymbol{\psi}^0_{0,2} \quad \boldsymbol{\psi}^0_{0,3} \quad \cdots \quad \boldsymbol{\psi}^0_{0,N-2} \quad \boldsymbol{\psi}^0_{0,N-1}\ \right]$$

$$\vdots$$

$$\left[\ \boldsymbol{\psi}^{j_{\max}-1}_{0,0} \quad \boldsymbol{\psi}^{j_{\max}-1}_{0,1}\ \right] \left[\ \boldsymbol{\psi}^{j_{\max}-1}_{1,0} \quad \boldsymbol{\psi}^{j_{\max}-1}_{1,1}\ \right] \cdots \left[\ \boldsymbol{\psi}^{j_{\max}-1}_{K^{j_{\max}-1}-1,0} \quad \boldsymbol{\psi}^{j_{\max}-1}_{K^{j_{\max}-1}-1,1}\ \right]$$

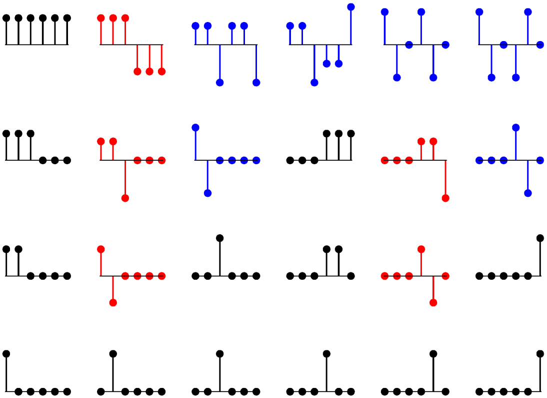
$$\left[\ \boldsymbol{\psi}^{j_{\max}}_{0,0}\ \right] \left[\ \boldsymbol{\psi}^{j_{\max}}_{1,0}\ \right] \left[\ \boldsymbol{\psi}^{j_{\max}}_{2,0}\ \right] \left[\ \boldsymbol{\psi}^{j_{\max}}_{3,0}\ \right] \cdots \left[\ \boldsymbol{\psi}^{j_{\max}}_{K^{j_{\max}}-2,0}\ \right] \quad \left[\ \boldsymbol{\psi}^{j_{\max}}_{K^{j_{\max}}-1,0}\ \right]$$

Remarks

- For an unweighted path graph, this yields a dictionary of Haar-Walsh functions
- As with the HGLET, we can select an orthonormal basis for the entire graph by taking the union of orthonormal bases on disjoint regions

Remarks

- For an unweighted path graph, this yields a dictionary of Haar-Walsh functions
- As with the HGLET, we can select an orthonormal basis for the entire graph by taking the union of orthonormal bases on disjoint regions



Remarks

- We can also reorder and regroup the vectors on each level of the GHWT dictionary according to their type (**scaling**, **Haar-like**, or **Walsh-like**)[1]

- This reorganization gives us *more options for choosing a good basis*

[1] Full details will be presented in a forthcoming article.

Remarks

- We can also reorder and regroup the vectors on each level of the GHWT dictionary according to their type (**scaling**, **Haar-like**, or **Walsh-like**)[1]

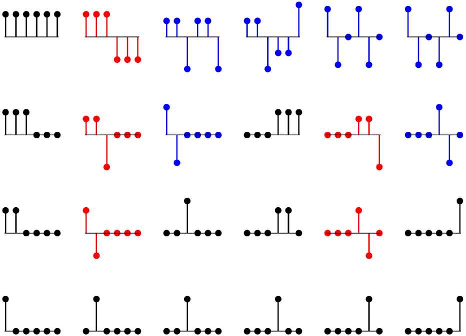


Figure: Default dictionary; i.e., coarse-to-fine

- This reorganization gives us *more options for choosing a good basis*

[1] Full details will be presented in a forthcoming article.

Remarks

- We can also reorder and regroup the vectors on each level of the GHWT dictionary according to their type (**scaling**, **Haar-like**, or **Walsh-like**)[1]

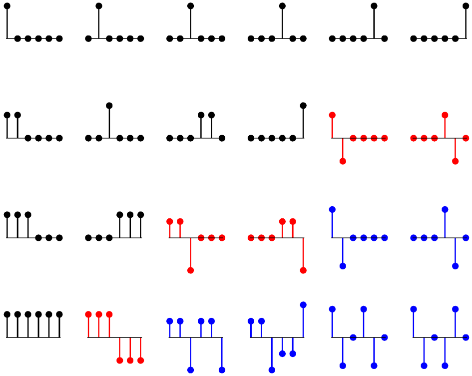


Figure: Reordered & regrouped dictionary; i.e., fine-to-coarse

- This reorganization gives us *more options for choosing a good basis*

[1] Full details will be presented in a forthcoming article.

Remarks

- We can also reorder and regroup the vectors on each level of the GHWT dictionary according to their type (**scaling**, **Haar-like**, or **Walsh-like**)[1]

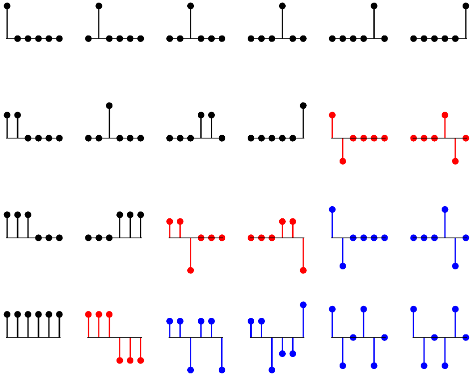


Figure: Reordered & regrouped dictionary; i.e., fine-to-coarse

- This reorganization gives us *more options for choosing a good basis*

[1] Full details will be presented in a forthcoming article.

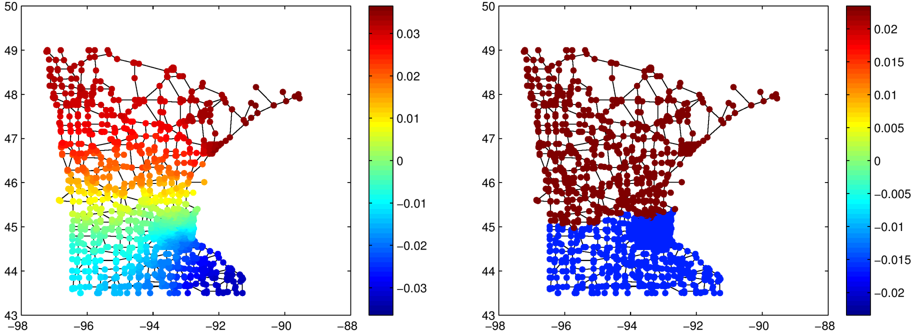
HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

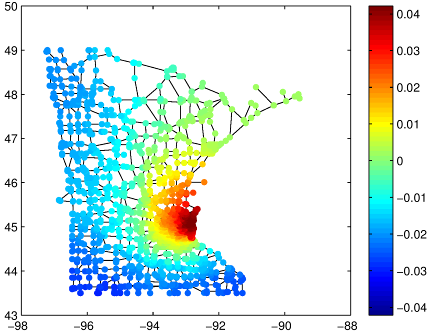
Level $j = 0$, Region $k = 0$, $l = 1$

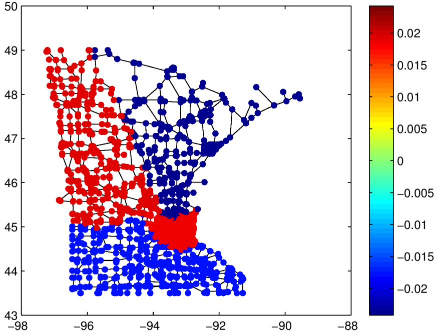


HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

Level $j = 0$, Region $k = 0$, $l = 2$

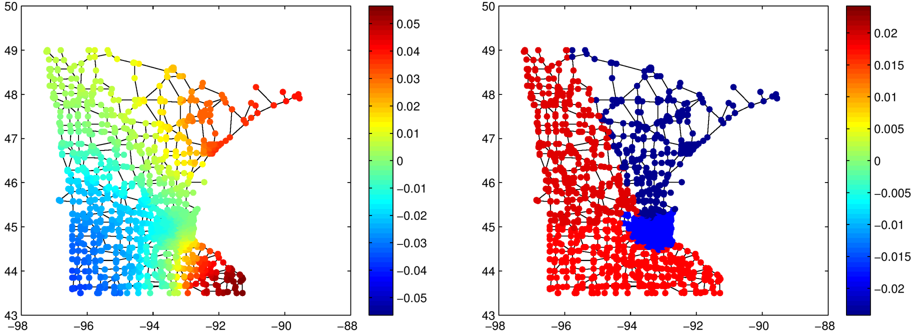




HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

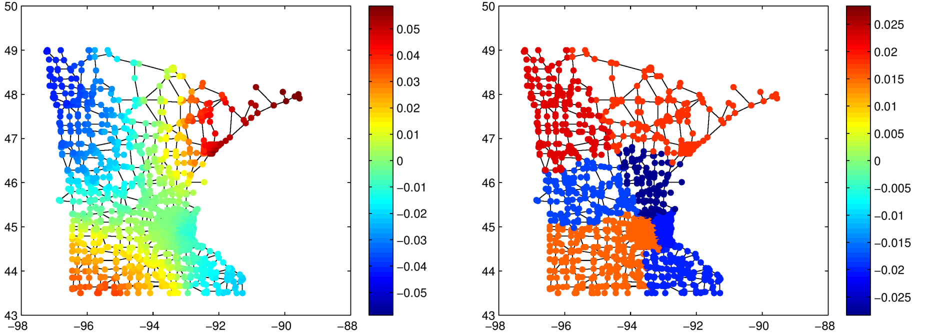
Level $j = 0$, Region $k = 0$, $l = 3$



HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

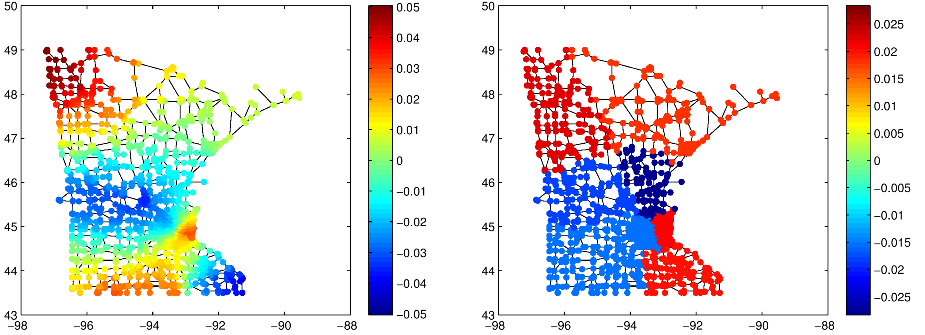
Level $j = 0$, Region $k = 0$, $l = 4$



HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

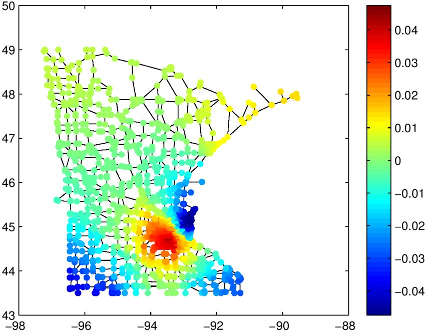
Level $j = 0$, Region $k = 0$, $l = 5$

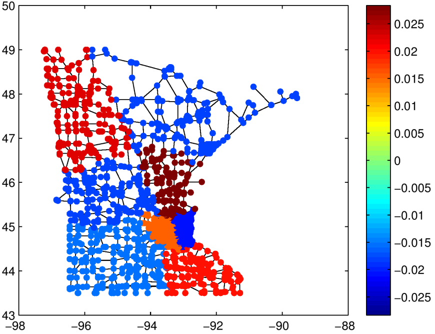


HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j=0$ is the coarsest scale, $j=14$ is the finest.)

Level $j=0$, Region $k=0$, $l=6$

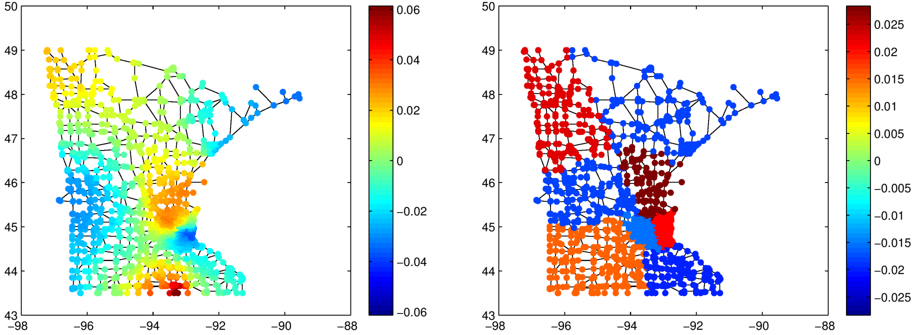




HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

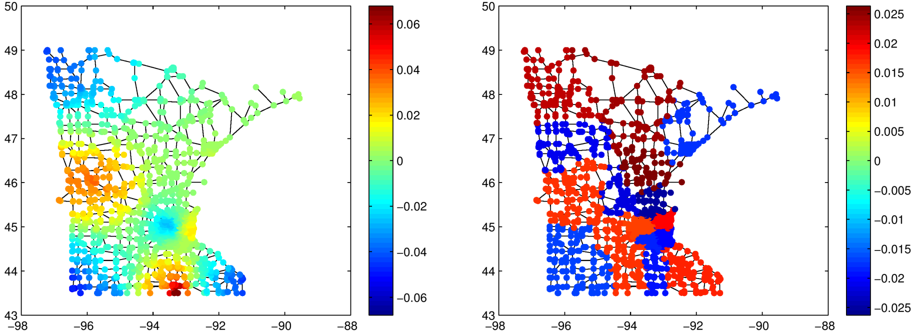
Level $j = 0$, Region $k = 0$, $l = 7$



HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

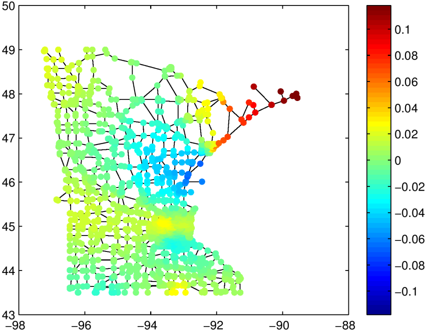
Level $j = 0$, Region $k = 0$, $l = 8$

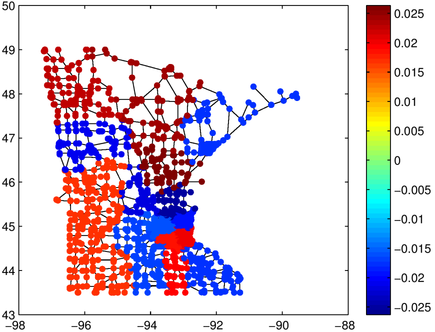


HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

Level $j = 0$, Region $k = 0$, $l = 9$

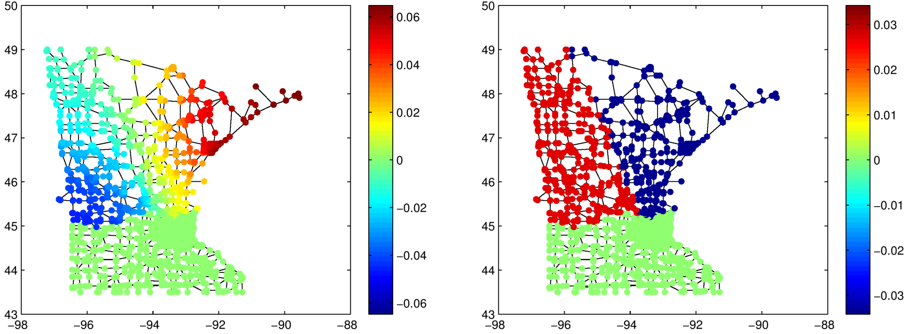




HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

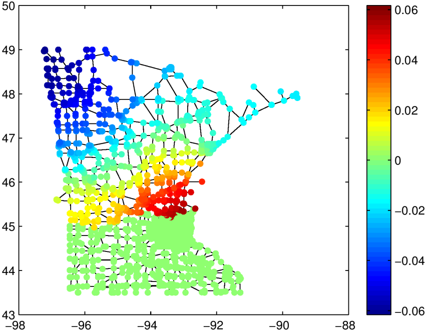
Level $j = 1$, Region $k = 0$, $l = 1$

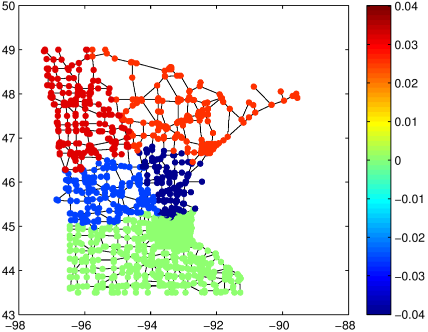


HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

Level $j = 1$, Region $k = 0$, $l = 2$

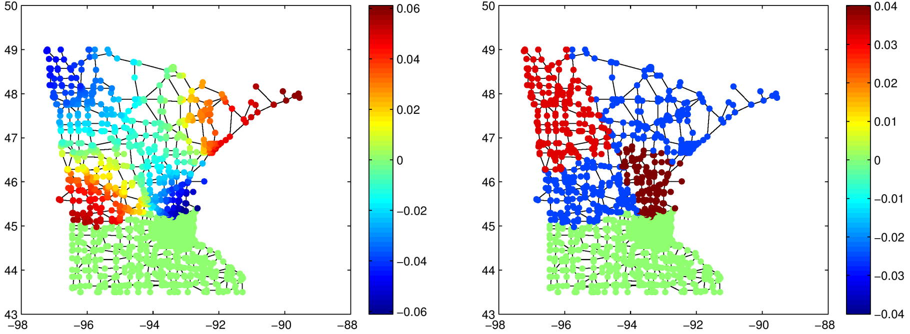




HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

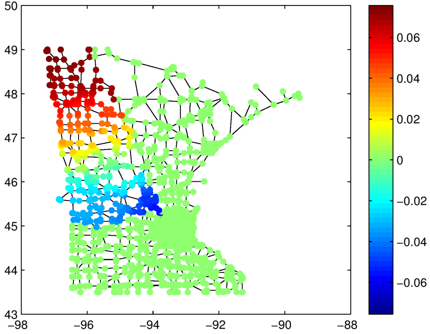
Level $j = 1$, Region $k = 0$, $l = 3$

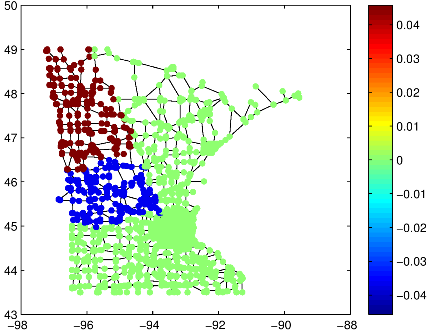


HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

Level $j = 2$, Region $k = 0$, $l = 1$

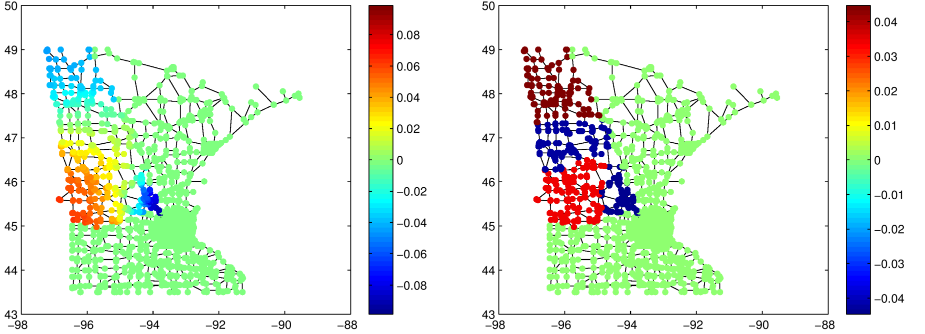




HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

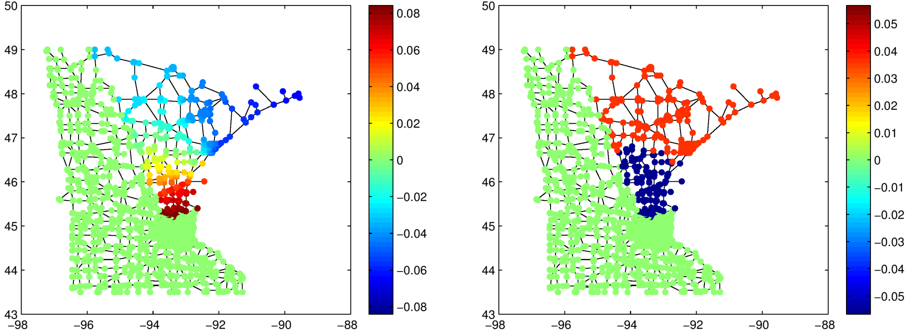
Level $j = 2$, Region $k = 0$, $l = 2$



HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

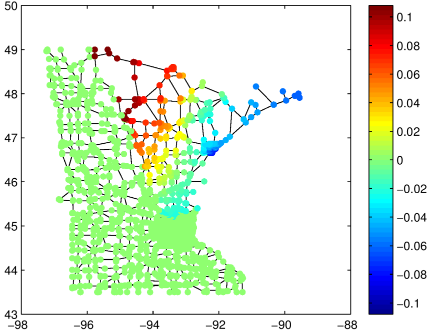
Level $j = 2$, Region $k = 1$, $l = 1$

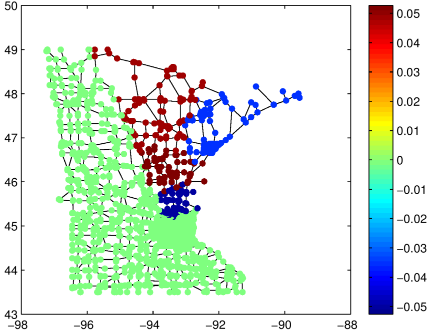


HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

Level $j = 2$, Region $k = 1$, $l = 2$

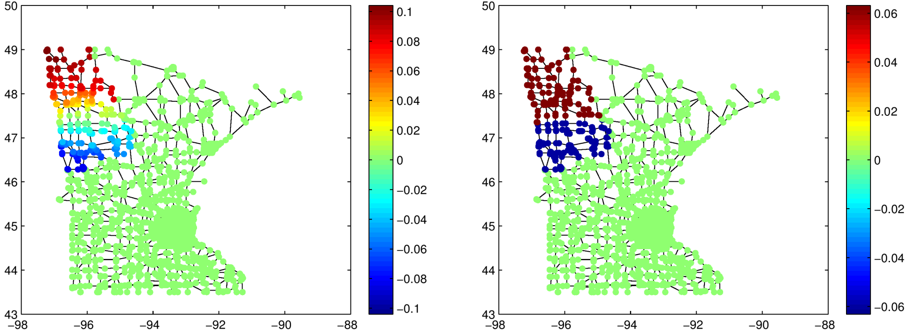




HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

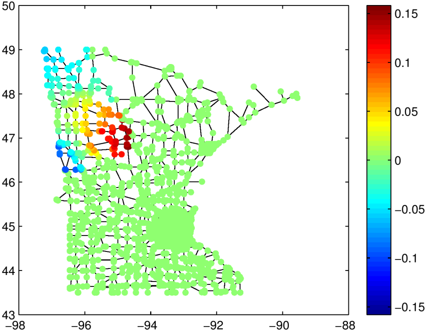
Level $j = 3$, Region $k = 0$, $l = 1$

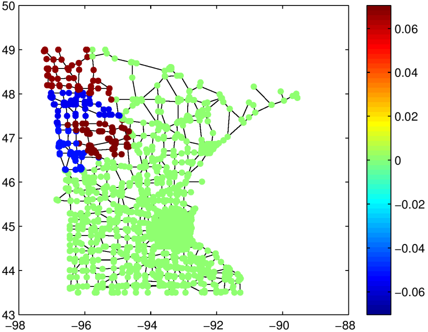


HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

Level $j = 3$, Region $k = 0$, $l = 2$

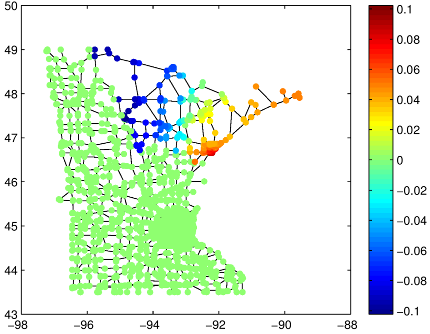


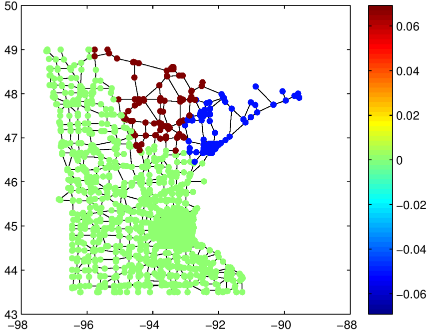


HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

Level $j = 3$, Region $k = 2$, $l = 1$

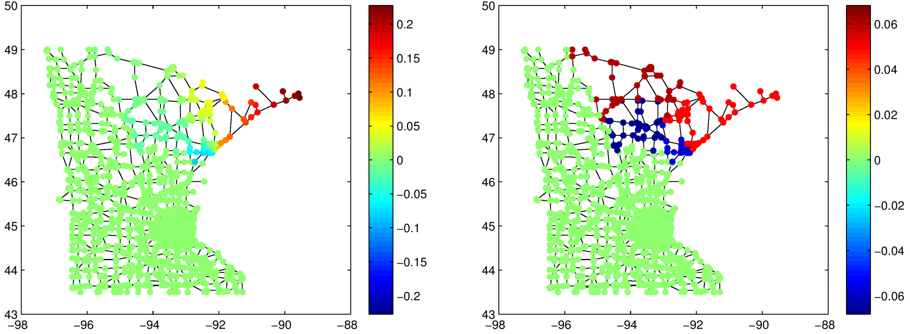




HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

Level $j = 3$, Region $k = 2$, $l = 2$



Computational Complexity: GHWT

	Computational Complexity	Run Time for MN[1]
HGLET (redundant)	$O(N^3)$	67 sec
GHWT (redundant)	$O(N^2)$	10 sec

[1] Computations performed on a personal laptop (4.00 GB RAM, 2.26 GHz), $N = 2640$ and `nnz(W)` $= 6604$.

Related Work

The following articles also discussed the Haar-like transform on graphs and trees, but *not the Walsh-Hadamard transform* on them:

1. A. D. Szlam, M. Maggioni, R. R. Coifman, and J. C. Bremer, Jr., "Diffusion-driven multiscale analysis on manifolds and graphs: top-down and bottom-up constructions," in *Wavelets XI* (M. Papadakis et al. eds.), *Proc. SPIE 5914*, Paper # 59141D, 2005.
2. F. Murtagh, "The Haar wavelet transform of a dendrogram," *J. Classification*, vol. 24, pp. 3–32, 2007.
3. A. Lee, B. Nadler, and L. Wasserman, "Treelets–an adaptive multi-scale basis for sparse unordered data," *Ann. Appl. Stat.*, vol. 2, pp. 435–471, 2008.
4. M. Gavish, B. Nadler, and R. Coifman, "Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning," in *Proc. 27th Intern. Conf. Machine Learning* (J. Fürnkranz et al. eds.), pp. 367–374, Omnipress, Haifa, 2010.

Coifman and Wickerhauser (1992) developed the best-basis algorithm as a means of selecting the basis from a dictionary of wavelet packets that is "best" for approximation/compression.

We generalize this approach, developing and implementing an algorithm for selecting the basis from the dictionary of HGLET / GHWT bases that is "best" for approximation.

As before, we require a cost functional $\mathscr{J}$. For example:

$$\mathscr{J}(\boldsymbol{x}) = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} = \texttt{norm(x,p)} \quad 0 < p \le 1$$

- For our approximation experiments in the following pages, we used $p = 0.1$.

Coifman and Wickerhauser (1992) developed the best-basis algorithm as a means of selecting the basis from a dictionary of wavelet packets that is "best" for approximation/compression.

We generalize this approach, developing and implementing an algorithm for selecting the basis from the dictionary of HGLET / GHWT bases that is "best" for approximation.

As before, we require a cost functional $\mathscr{J}$. For example:

$$\mathscr{J}(\boldsymbol{x}) = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} = \texttt{norm(x,p)} \quad 0 < p \leq 1$$

- For our approximation experiments in the following pages, we used $p = 0.1$.

Coifman and Wickerhauser (1992) developed the best-basis algorithm as a means of selecting the basis from a dictionary of wavelet packets that is "best" for approximation/compression.

We generalize this approach, developing and implementing an algorithm for selecting the basis from the dictionary of HGLET / GHWT bases that is "best" for approximation.

As before, we require a cost functional $\mathscr{J}$. For example:

$$\mathscr{J}(\boldsymbol{x}) = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} = \texttt{norm(x,p)} \quad 0 < p \le 1$$

- For our approximation experiments in the following pages, we used $p = 0.1$.

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N^0_0-1} \end{bmatrix}$$
$$d^0_{0,0} \quad d^0_{0,1} \quad d^0_{0,2} \quad \cdots \quad d^0_{0,N^0_0-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N^1_1-1} \end{bmatrix}$$
$$d^1_{0,0} \quad d^1_{0,1} \quad d^1_{0,2} \quad \cdots \quad d^1_{0,N^1_0-1} \qquad d^1_{1,0} \quad d^1_{1,1} \quad d^1_{1,2} \quad \cdots \quad d^1_{1,N^1_1-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{0,0}\, \boldsymbol{\phi}^2_{0,1} \cdots \boldsymbol{\phi}^2_{0,N^2_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{1,0}\, \boldsymbol{\phi}^2_{1,1} \cdots \boldsymbol{\phi}^2_{1,N^2_1-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{2,0}\, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N^2_2-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{3,0}\, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N^2_3-1} \end{bmatrix}$$
$$d^2_{0,0}\; d^2_{0,1} \cdots d^2_{0,N^2_0-1} \quad d^2_{1,0}\; d^2_{1,1} \cdots d^2_{1,N^2_1-1} \quad d^2_{2,0}\; d^2_{2,1} \cdots d^2_{2,N^2_2-1} \quad d^2_{3,0}\; d^2_{3,1} \cdots d^2_{3,N^2_3-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N^0_0-1} \end{bmatrix}$$
$$d^0_{0,0} \quad d^0_{0,1} \quad d^0_{0,2} \quad \cdots \quad d^0_{0,N^0_0-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N^1_1-1} \end{bmatrix}$$
$$d^1_{0,0} \quad d^1_{0,1} \quad d^1_{0,2} \quad \cdots \quad d^1_{0,N^1_0-1} \qquad d^1_{1,0} \quad d^1_{1,1} \quad d^1_{1,2} \quad \cdots \quad d^1_{1,N^1_1-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{0,0}\, \boldsymbol{\phi}^2_{0,1} \cdots \boldsymbol{\phi}^2_{0,N^2_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{1,0}\, \boldsymbol{\phi}^2_{1,1} \cdots \boldsymbol{\phi}^2_{1,N^2_1-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{2,0}\, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N^2_2-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{3,0}\, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N^2_3-1} \end{bmatrix}$$
$$d^2_{0,0}\, d^2_{0,1} \cdots d^2_{0,N^2_0-1} \quad d^2_{1,0}\, d^2_{1,1} \cdots d^2_{1,N^2_1-1} \quad d^2_{2,0}\, d^2_{2,1} \cdots d^2_{2,N^2_2-1} \quad d^2_{3,0}\, d^2_{3,1} \cdots d^2_{3,N^2_3-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N^0_0-1} \end{bmatrix}$$
$$d^0_{0,0} \quad d^0_{0,1} \quad d^0_{0,2} \quad \cdots \quad d^0_{0,N^0_0-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N^1_1-1} \end{bmatrix}$$
$$d^1_{0,0} \quad d^1_{0,1} \quad d^1_{0,2} \quad \cdots \quad d^1_{0,N^1_0-1} \qquad d^1_{1,0} \quad d^1_{1,1} \quad d^1_{1,2} \quad \cdots \quad d^1_{1,N^1_1-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{0,0}\, \boldsymbol{\phi}^2_{0,1} \cdots \boldsymbol{\phi}^2_{0,N^2_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{1,0}\, \boldsymbol{\phi}^2_{1,1} \cdots \boldsymbol{\phi}^2_{1,N^2_1-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{2,0}\, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N^2_2-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{3,0}\, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N^2_3-1} \end{bmatrix}$$
$$d^2_{0,0}\; d^2_{0,1} \cdots d^2_{0,N^2_0-1} \quad d^2_{1,0}\; d^2_{1,1} \cdots d^2_{1,N^2_1-1} \quad d^2_{2,0}\; d^2_{2,1} \cdots d^2_{2,N^2_2-1} \quad d^2_{3,0}\; d^2_{3,1} \cdots d^2_{3,N^2_3-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N^0_0-1} \end{bmatrix}$$
$$d^0_{0,0} \quad d^0_{0,1} \quad d^0_{0,2} \quad \cdots \quad d^0_{0,N^0_0-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N^1_1-1} \end{bmatrix}$$
$$d^1_{0,0} \quad d^1_{0,1} \quad d^1_{0,2} \quad \cdots \quad d^1_{0,N^1_0-1} \qquad d^1_{1,0} \quad d^1_{1,1} \quad d^1_{1,2} \quad \cdots \quad d^1_{1,N^1_1-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{2,0} \, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N^2_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} \, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N^2_3-1} \end{bmatrix}$$
$$d^2_{2,0} \; d^2_{2,1} \cdots d^2_{2,N^2_2-1} \qquad d^2_{3,0} \; d^2_{3,1} \cdots d^2_{3,N^2_3-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N^0_0-1} \end{bmatrix}$$
$$d^0_{0,0} \quad d^0_{0,1} \quad d^0_{0,2} \quad \cdots \quad d^0_{0,N^0_0-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N^1_1-1} \end{bmatrix}$$
$$d^1_{0,0} \quad d^1_{0,1} \quad d^1_{0,2} \quad \cdots \quad d^1_{0,N^1_0-1} \qquad d^1_{1,0} \quad d^1_{1,1} \quad d^1_{1,2} \quad \cdots \quad d^1_{1,N^1_1-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{2,0} \, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N^2_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} \, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N^2_3-1} \end{bmatrix}$$
$$d^2_{2,0} \; d^2_{2,1} \cdots d^2_{2,N^2_2-1} \qquad d^2_{3,0} \; d^2_{3,1} \cdots d^2_{3,N^2_3-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N^0_0-1} \end{bmatrix}$$
$$d^0_{0,0} \quad d^0_{0,1} \quad d^0_{0,2} \quad \cdots \quad d^0_{0,N^0_0-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{bmatrix}$$
$$d^1_{0,0} \quad d^1_{0,1} \quad d^1_{0,2} \quad \cdots \quad d^1_{0,N^1_0-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{2,0} \, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N^2_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} \, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N^2_3-1} \end{bmatrix}$$
$$d^2_{2,0} \; d^2_{2,1} \cdots d^2_{2,N^2_2-1} \qquad d^2_{3,0} \; d^2_{3,1} \cdots d^2_{3,N^2_3-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N^0_0-1} \end{bmatrix}$$
$$d^0_{0,0} \quad d^0_{0,1} \quad d^0_{0,2} \quad \cdots \quad d^0_{0,N^0_0-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{bmatrix}$$
$$d^1_{0,0} \quad d^1_{0,1} \quad d^1_{0,2} \quad \cdots \quad d^1_{0,N^1_0-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{2,0} \, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N^2_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} \, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N^2_3-1} \end{bmatrix}$$
$$d^2_{2,0} \; d^2_{2,1} \cdots d^2_{2,N^2_2-1} \qquad d^2_{3,0} \; d^2_{3,1} \cdots d^2_{3,N^2_3-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{bmatrix}$$

$$d^1_{0,0} \quad d^1_{0,1} \quad d^1_{0,2} \quad \cdots \quad d^1_{0,N^1_0-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{2,0} \, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N^2_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} \, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N^2_3-1} \end{bmatrix}$$

$$d^2_{2,0} \; d^2_{2,1} \cdots d^2_{2,N^2_2-1} \quad d^2_{3,0} \; d^2_{3,1} \cdots d^2_{3,N^2_3-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{bmatrix}$$
$$d^1_{0,0} \quad d^1_{0,1} \quad d^1_{0,2} \quad \cdots \quad d^1_{0,N^1_0-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{2,0}\, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N^2_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0}\, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N^2_3-1} \end{bmatrix}$$
$$d^2_{2,0}\; d^2_{2,1} \cdots d^2_{2,N^2_2-1} \qquad d^2_{3,0}\; d^2_{3,1} \cdots d^2_{3,N^2_3-1}$$

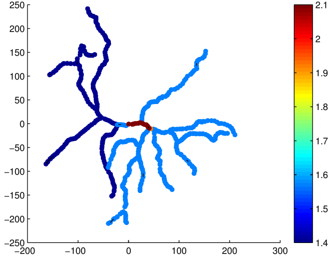
According to cost functional $\mathscr{J}$, this is the best basis for approximation.

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{bmatrix}$$
$$d^1_{0,0} \quad d^1_{0,1} \quad d^1_{0,2} \quad \cdots \quad d^1_{0,N^1_0-1}$$

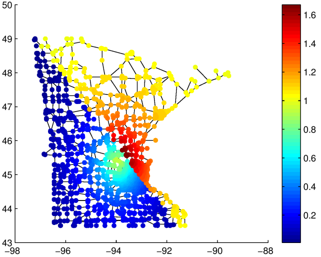
$$\begin{bmatrix} \boldsymbol{\phi}^2_{2,0}\, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N^2_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0}\, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N^2_3-1} \end{bmatrix}$$
$$d^2_{2,0}\; d^2_{2,1} \cdots d^2_{2,N^2_2-1} \qquad d^2_{3,0}\; d^2_{3,1} \cdots d^2_{3,N^2_3-1}$$

According to cost functional $\mathscr{J}$, this is the best basis for approximation.

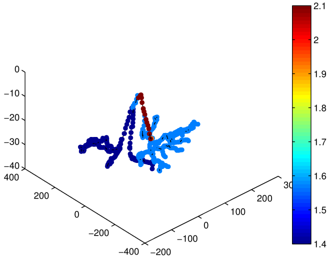
- With the GHWT bases, we run the best-basis algorithm on both the default (coarse-to-fine) dictionary and the reorganized (fine-to-coarse) dictionary and then compare the cost of the 2 bases to determine the best-basis.



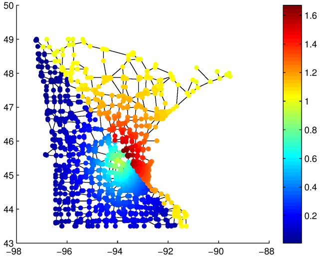
(a) Thickness data on a dendritic tree



(b) A mutilated Gaussian on the MN road network

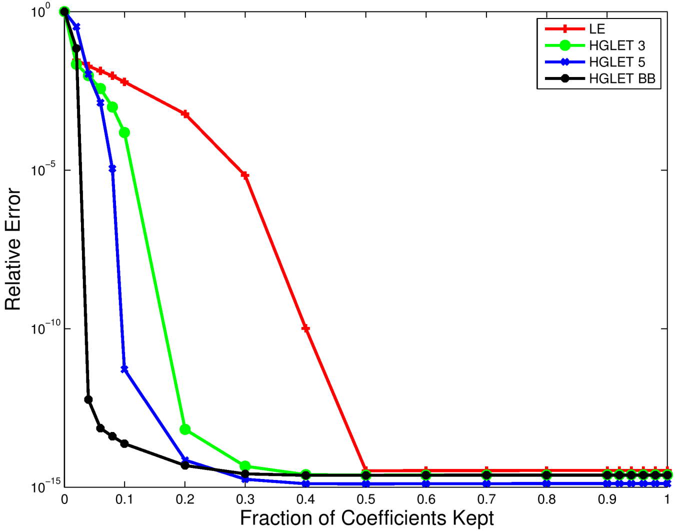


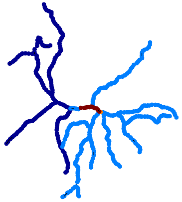
(a) Thickness data on a dendritic tree



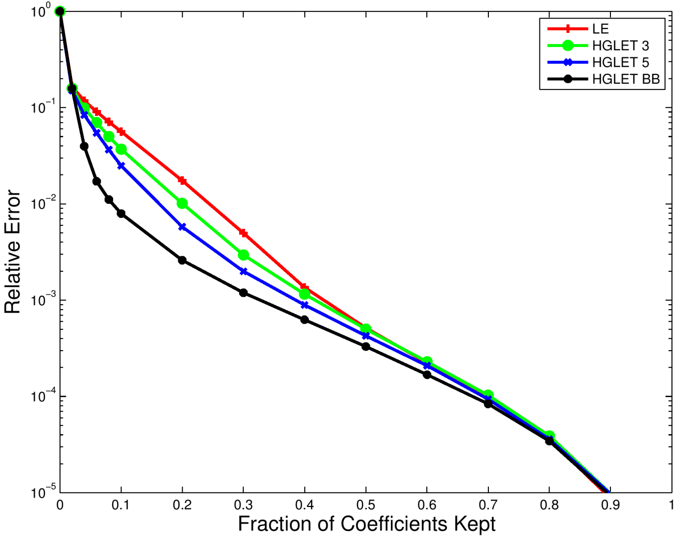
(b) A mutilated Gaussian on the MN road network

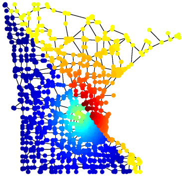
HGLET on Dendrite (weights = inv. Euclidean dist.)



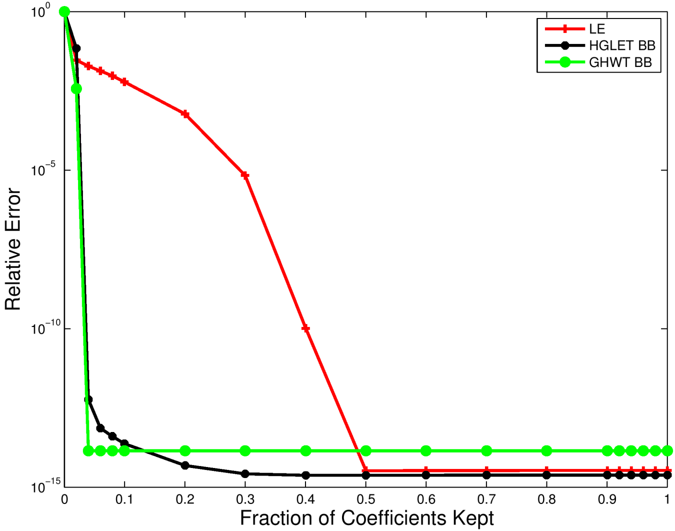


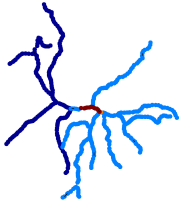
HGLET on MN Mutilated Gaussian (weights = inv. Euclidean dist.)



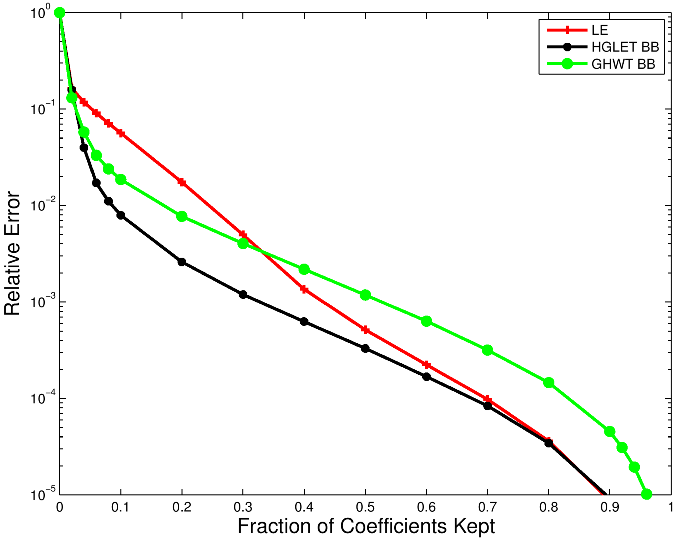


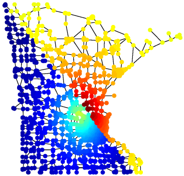
GHWT vs. HGLET on Dendrite





GHWT vs. HGLET on MN Mutilated Gaussian





Discussion of Approximation Results

- From the HGLET plots, we see that HGLET best-basis > HGLET Level 5 > HGLET Level 3 > Laplacian eigenvectors (HGLET Level 0)
- The HGLET best-basis performs the best on the MN Multilated Gaussian dataset while the GHWT best-basis outperformed the others on the Dendrite dataset
- These performances make a strong case for using localized basis vectors on *multiple scales*
- Also, these indicate that the *smoothness* of the basis vectors matters depending on the smoothness inherent in data

Discussion of Approximation Results

- From the HGLET plots, we see that HGLET best-basis > HGLET Level 5 > HGLET Level 3 > Laplacian eigenvectors (HGLET Level 0)
- The HGLET best-basis performs the best on the MN Multilated Gaussian dataset while the GHWT best-basis outperformed the others on the Dendrite dataset
- These performances make a strong case for using localized basis vectors on *multiple scales*
- Also, these indicate that the *smoothness* of the basis vectors matters depending on the smoothness inherent in data

Discussion of Approximation Results

- From the HGLET plots, we see that HGLET best-basis > HGLET Level 5 > HGLET Level 3 > Laplacian eigenvectors (HGLET Level 0)
- The HGLET best-basis performs the best on the MN Multilated Gaussian dataset while the GHWT best-basis outperformed the others on the Dendrite dataset
- These performances make a strong case for using localized basis vectors on *multiple scales*
- Also, these indicate that the *smoothness* of the basis vectors matters depending on the smoothness inherent in data

Discussion of Approximation Results

- From the HGLET plots, we see that HGLET best-basis > HGLET Level 5 > HGLET Level 3 > Laplacian eigenvectors (HGLET Level 0)
- The HGLET best-basis performs the best on the MN Multilated Gaussian dataset while the GHWT best-basis outperformed the others on the Dendrite dataset
- These performances make a strong case for using localized basis vectors on *multiple scales*
- Also, these indicate that the *smoothness* of the basis vectors matters depending on the smoothness inherent in data

Summary

- We developed multiscale basis dictionaries on graphs and networks: HGLET and GHWT. We also developed a corresponding best-basis algorithm.
- The HGLET is a direct generalization of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
- The GHWT is a generalization of the *Haar Transform* and the *Walsh-Hadamard Transform*.
- Both of these transforms allow us to choose an orthonormal basis most suitable for the task at hand, e.g., approximation, classification, regression, . . .
- They may also be useful for regularly-sampled signals, e.g., can deal with signals of non-dyadic length; adaptive segmentation, . . .
- Developing a *true* generalization of smoother wavelet and wavelet packet transforms is more challenging due to the difficulty of the notion of the *frequency domain* of a given graph.

Summary

- We developed multiscale basis dictionaries on graphs and networks: HGLET and GHWT. We also developed a corresponding best-basis algorithm.
- The HGLET is a direct generalization of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
- The GHWT is a generalization of the *Haar Transform* and the *Walsh-Hadamard Transform*.
- Both of these transforms allow us to choose an orthonormal basis most suitable for the task at hand, e.g., approximation, classification, regression, . . .
- They may also be useful for regularly-sampled signals, e.g., can deal with signals of non-dyadic length; adaptive segmentation, . . .
- Developing a *true* generalization of smoother wavelet and wavelet packet transforms is more challenging due to the difficulty of the notion of the *frequency domain* of a given graph.

Summary

- We developed multiscale basis dictionaries on graphs and networks: HGLET and GHWT. We also developed a corresponding best-basis algorithm.
- The HGLET is a direct generalization of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
- The GHWT is a generalization of the *Haar Transform* and the *Walsh-Hadamard Transform*.
- Both of these transforms allow us to choose an orthonormal basis most suitable for the task at hand, e.g., approximation, classification, regression, . . .
- They may also be useful for regularly-sampled signals, e.g., can deal with signals of non-dyadic length; adaptive segmentation, . . .
- Developing a *true* generalization of smoother wavelet and wavelet packet transforms is more challenging due to the difficulty of the notion of the *frequency domain* of a given graph.

Summary

- We developed multiscale basis dictionaries on graphs and networks: HGLET and GHWT. We also developed a corresponding best-basis algorithm.
- The HGLET is a direct generalization of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
- The GHWT is a generalization of the *Haar Transform* and the *Walsh-Hadamard Transform*.
- Both of these transforms allow us to choose an orthonormal basis most suitable for the task at hand, e.g., approximation, classification, regression, . . .
- They may also be useful for regularly-sampled signals, e.g., can deal with signals of non-dyadic length; adaptive segmentation, . . .
- Developing a *true* generalization of smoother wavelet and wavelet packet transforms is more challenging due to the difficulty of the notion of the *frequency domain* of a given graph.

Summary

- We developed multiscale basis dictionaries on graphs and networks: HGLET and GHWT. We also developed a corresponding best-basis algorithm.
- The HGLET is a direct generalization of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
- The GHWT is a generalization of the *Haar Transform* and the *Walsh-Hadamard Transform*.
- Both of these transforms allow us to choose an orthonormal basis most suitable for the task at hand, e.g., approximation, classification, regression, . . .
- They may also be useful for regularly-sampled signals, e.g., can deal with signals of non-dyadic length; adaptive segmentation, . . .
- Developing a *true* generalization of smoother wavelet and wavelet packet transforms is more challenging due to the difficulty of the notion of the *frequency domain* of a given graph.

Summary

- We developed multiscale basis dictionaries on graphs and networks: HGLET and GHWT. We also developed a corresponding best-basis algorithm.
- The HGLET is a direct generalization of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
- The GHWT is a generalization of the *Haar Transform* and the *Walsh-Hadamard Transform*.
- Both of these transforms allow us to choose an orthonormal basis most suitable for the task at hand, e.g., approximation, classification, regression, . . .
- They may also be useful for regularly-sampled signals, e.g., can deal with signals of non-dyadic length; adaptive segmentation, . . .
- Developing a *true* generalization of smoother wavelet and wavelet packet transforms is more challenging due to the difficulty of the notion of the *frequency domain* of a given graph.

Future Work

- Perform classification experiments and compare the results using HGLET and GHWT.
- Explore other methods for graph partitioning:
  - Allow for splitting of a region into an arbitrary number of subregions;
  - Consider a bottom-up clustering method, rather than a top-down partitioning method;
  - Incorporate the *diffuse interface model* and the minimization of the *Ginsburg-Landau functional* proposed by Bertozzi and Flenner (2012).

Future Work

- Perform classification experiments and compare the results using HGLET and GHWT.
- Explore other methods for graph partitioning:
  - Allow for splitting of a region into an arbitrary number of subregions;
  - Consider a bottom-up clustering method, rather than a top-down partitioning method;
  - Incorporate the *diffuse interface model* and the minimization of the *Ginsburg-Landau functional* proposed by Bertozzi and Flenner (2012).

References & Acknowledgments

- http://www.math.ucdavis.edu/~saito/courses/HarmGraph/ contains my course slides and useful information on "Harmonic Analysis on Graphs and Networks"
- Also visit http://www.math.ucdavis.edu/~saito/publications/ for various related publications including:

  - N. Saito: "Data analysis and representation using eigenfunctions of Laplacian on a general domain," *Applied & Computational Harmonic Analysis*, vol. 25, no. 1, pp. 68–97, 2008.
  - N. Saito & E. Woei: "Analysis of neuronal dendrite patterns using eigenvalues of graph Laplacians," *Japan SIAM Letters*, vol. 1, pp. 13–16, 2009.
  - Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra & Its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
  - J. Irion & N. Saito: "Hierarchical graph Laplacian eigen transforms," *Japan SIAM Letters*, to appear, 2014.

This research was partially supported by the grants received from the *Office of Naval Research* and the *National Defense Science and Engineering Graduate Fellowship*.