Natural Constructions of Smooth Graph Wavelet Packets

Naoki Saito
in collaboration with
Haotian Li (UCD) and Alex Cloninger (UCSD)

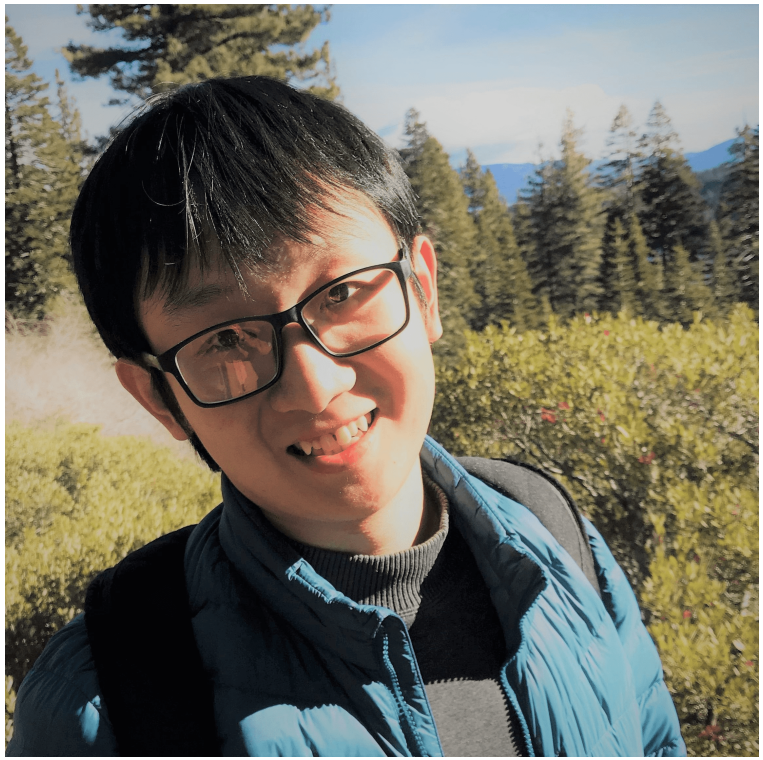
Department of Mathematics
University of California, Davis

Virtual Workshop on "Time–Frequency Frames and Their Applications"
Research Institute for Mathematical Sciences, Kyoto University, Japan
October 20, 2020

Outline

Acknowledgment

- NSF Grants: DMS-1819222, DMS-1912747, CCF-1934568, DMS-2012266
- ONR Grants: N00014-16-1-2255, N00014-20-1-2381
- Russell Sage Foundation Grant #2196



Haotian Li (UCD)

Alex Cloninger (UCSD)

Outline

Motivations

- We are interested in generating a ***smooth*** *wavelet packet dictionary* on a given graph, which truly generalizes its classical counter part.
- We have already developed the *Hierarchical Graph Laplacian Eigen Transform* (HGLET) [J. Irion & N. Saito, 2014a]; the *Generalized Haar-Walsh Transform* (GHWT) [J. Irion & N. Saito, 2014b]; and its extension *eGHWT* [Y. Shao & N. Saito, 2019].
- HGLET is a graph version of the *Hierarchical Block Discrete Cosine Transform* (BDCT) while GHWT/eGHWT are graph versions of the *Haar-Walsh wavelet packet transform* [R. Coifman & Y. Meyer, 1989]/the *adapted time-frequency tilings* [C. Thiele & L. Villemoes, 1996].
- All of these methods require the *hierarchical bipartition tree* of an input graph G, and the support of each basis vector is restricted within the partition pattern. That is, for a given pair of mutually exclusive subgraphs of G, the support of the basis vectors of one of such subgraphs does *not smoothly crossover* to the other subgraph.

Motivations

- We are interested in generating a ***smooth*** *wavelet packet dictionary* on a given graph, which truly generalizes its classical counter part.
- We have already developed the *Hierarchical Graph Laplacian Eigen Transform* (HGLET) [J. Irion & N. Saito, 2014a]; the *Generalized Haar-Walsh Transform* (GHWT) [J. Irion & N. Saito, 2014b]; and its extension *eGHWT* [Y. Shao & N. Saito, 2019].
- HGLET is a graph version of the *Hierarchical Block Discrete Cosine Transform* (BDCT) while GHWT/eGHWT are graph versions of the *Haar-Walsh wavelet packet transform* [R. Coifman & Y. Meyer, 1989]/the *adapted time-frequency tilings* [C. Thiele & L. Villemoes, 1996].
- All of these methods require the *hierarchical bipartition tree* of an input graph G, and the support of each basis vector is restricted within the partition pattern. That is, for a given pair of mutually exclusive subgraphs of G, the support of the basis vectors of one of such subgraphs does *not smoothly crossover* to the other subgraph.

Motivations

- We are interested in generating a ***smooth*** *wavelet packet dictionary* on a given graph, which truly generalizes its classical counter part.
- We have already developed the *Hierarchical Graph Laplacian Eigen Transform* (HGLET) [J. Irion & N. Saito, 2014a]; the *Generalized Haar-Walsh Transform* (GHWT) [J. Irion & N. Saito, 2014b]; and its extension *eGHWT* [Y. Shao & N. Saito, 2019].
- HGLET is a graph version of the *Hierarchical Block Discrete Cosine Transform* (BDCT) while GHWT/eGHWT are graph versions of the *Haar-Walsh wavelet packet transform* [R. Coifman & Y. Meyer, 1989]/the *adapted time-frequency tilings* [C. Thiele & L. Villemoes, 1996].
- All of these methods require the *hierarchical bipartition tree* of an input graph G, and the support of each basis vector is restricted within the partition pattern. That is, for a given pair of mutually exclusive subgraphs of G, the support of the basis vectors of one of such subgraphs does *not smoothly crossover* to the other subgraph.

Motivations

- We are interested in generating a ***smooth*** *wavelet packet dictionary* on a given graph, which truly generalizes its classical counter part.
- We have already developed the *Hierarchical Graph Laplacian Eigen Transform* (HGLET) [J. Irion & N. Saito, 2014a]; the *Generalized Haar-Walsh Transform* (GHWT) [J. Irion & N. Saito, 2014b]; and its extension *eGHWT* [Y. Shao & N. Saito, 2019].
- HGLET is a graph version of the *Hierarchical Block Discrete Cosine Transform* (BDCT) while GHWT/eGHWT are graph versions of the *Haar-Walsh wavelet packet transform* [R. Coifman & Y. Meyer, 1989]/the *adapted time-frequency tilings* [C. Thiele & L. Villemoes, 1996].
- All of these methods require the *hierarchical bipartition tree* of an input graph G, and the support of each basis vector is restricted within the partition pattern. That is, for a given pair of mutually exclusive subgraphs of G, the support of the basis vectors of one of such subgraphs does *not smoothly crossover* to the other subgraph.

Motivations . . .

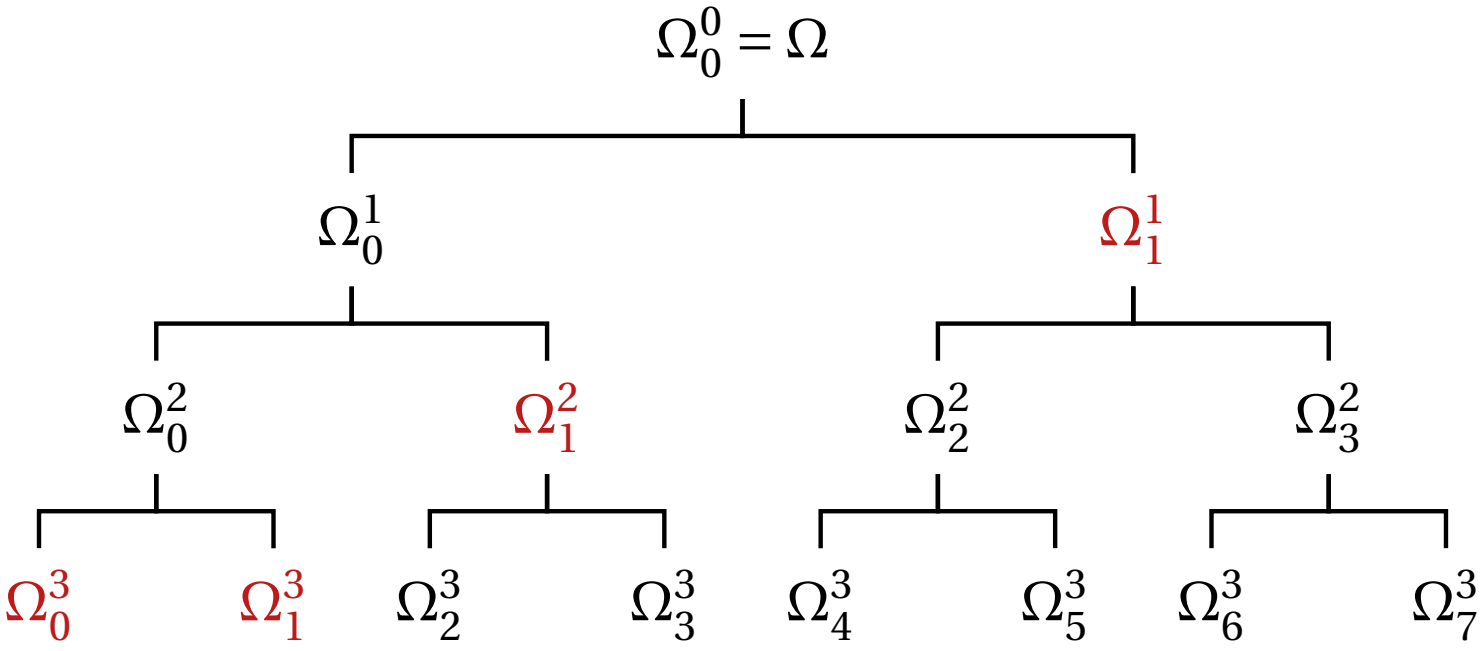


Figure: A binary partition tree of the input space Ω

- 1D Hierarchical BDCT $\Longleftrightarrow \Omega = [0,1]$ on the spatial axis
- HGLET, GHWT, eGHWT $\Longleftrightarrow \Omega = G$, a graph
- Classical wavelet packets $\Longleftrightarrow \Omega = [0, \frac{1}{2}]$ on the *frequency* axis
- *Smooth graph wavelet packets* $\Longleftrightarrow \Omega =$??

Motivations . . .

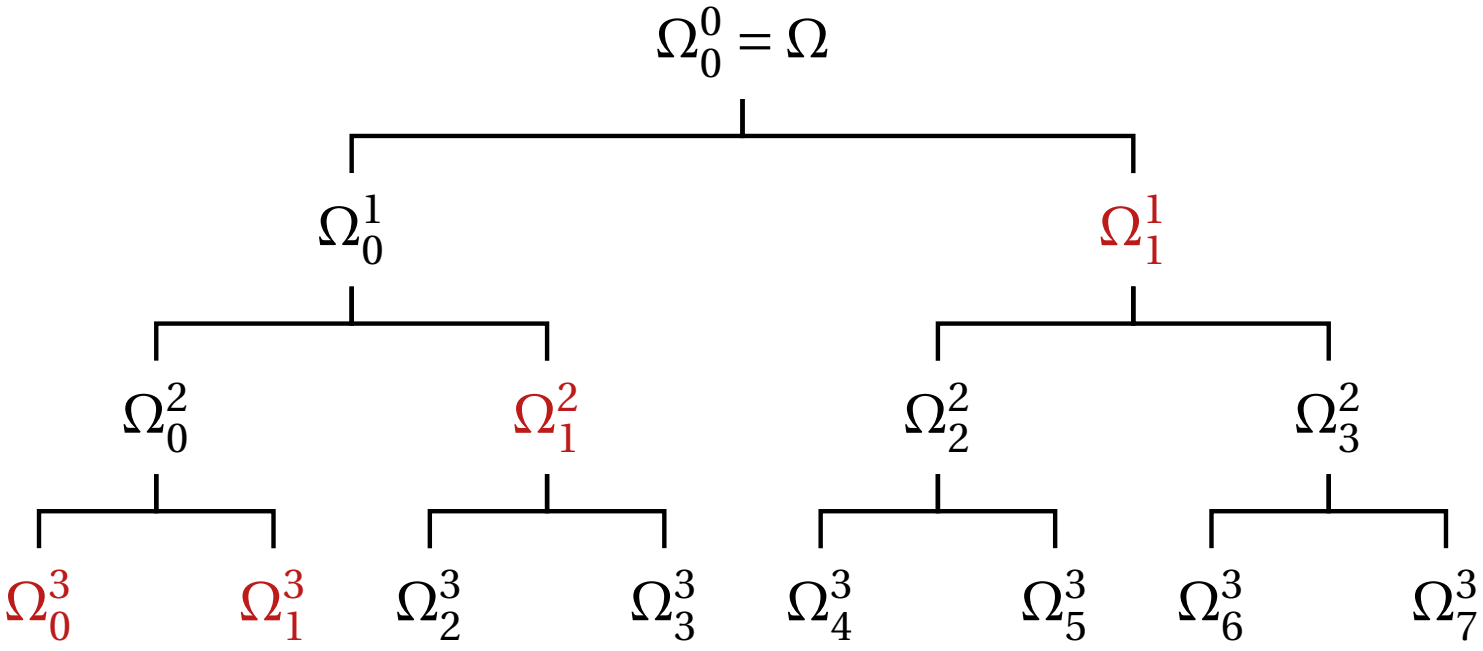


Figure: A binary partition tree of the input space Ω

- 1D Hierarchical BDCT $\Longleftrightarrow \Omega = [0,1]$ on the spatial axis
- HGLET, GHWT, eGHWT $\Longleftrightarrow \Omega = G$, a graph
- Classical wavelet packets $\Longleftrightarrow \Omega = [0, \frac{1}{2}]$ on the *frequency* axis
- *Smooth graph wavelet packets* $\Longleftrightarrow \Omega =$??

Motivations . . .

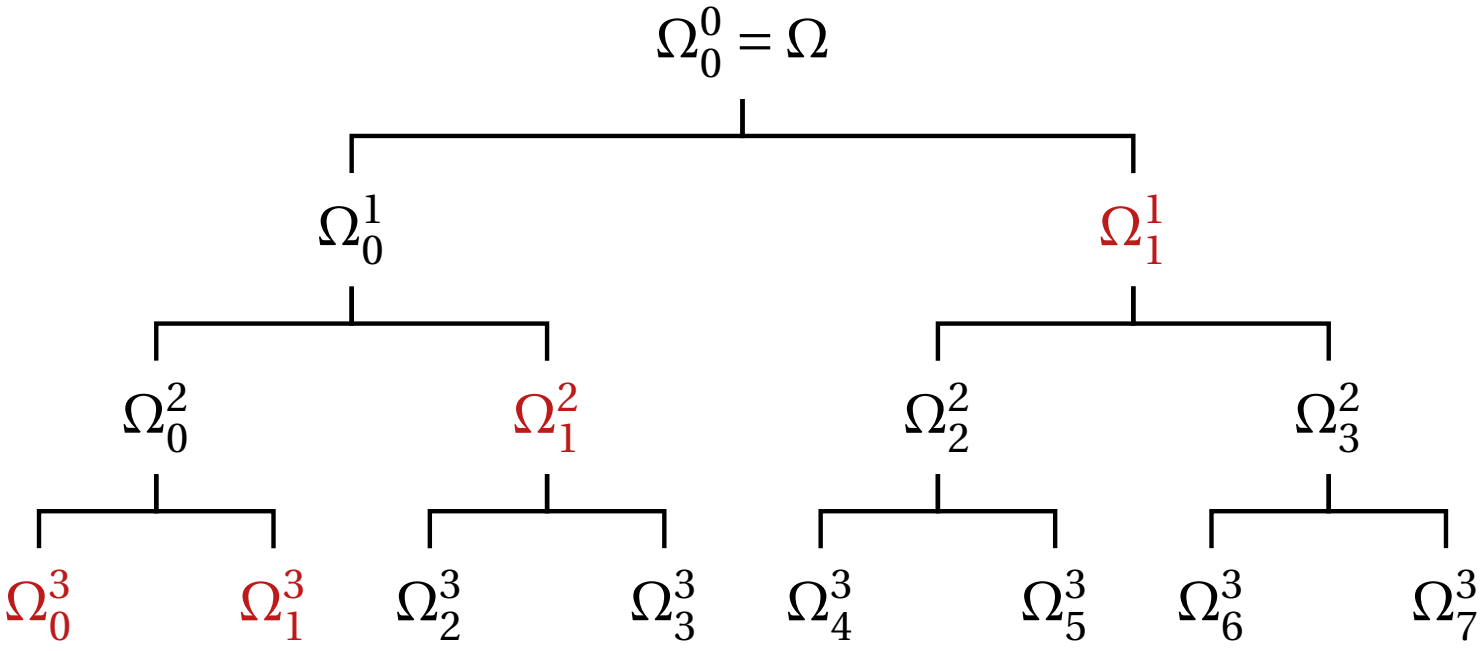


Figure: A binary partition tree of the input space Ω

- 1D Hierarchical BDCT $\Longleftrightarrow \Omega = [0,1]$ on the spatial axis
- HGLET, GHWT, eGHWT $\Longleftrightarrow \Omega = G$, a graph
- Classical wavelet packets $\Longleftrightarrow \Omega = [0, \frac{1}{2}]$ on the *frequency* axis
- *Smooth graph wavelet packets* $\Longleftrightarrow \Omega = ??$

Motivations . . .

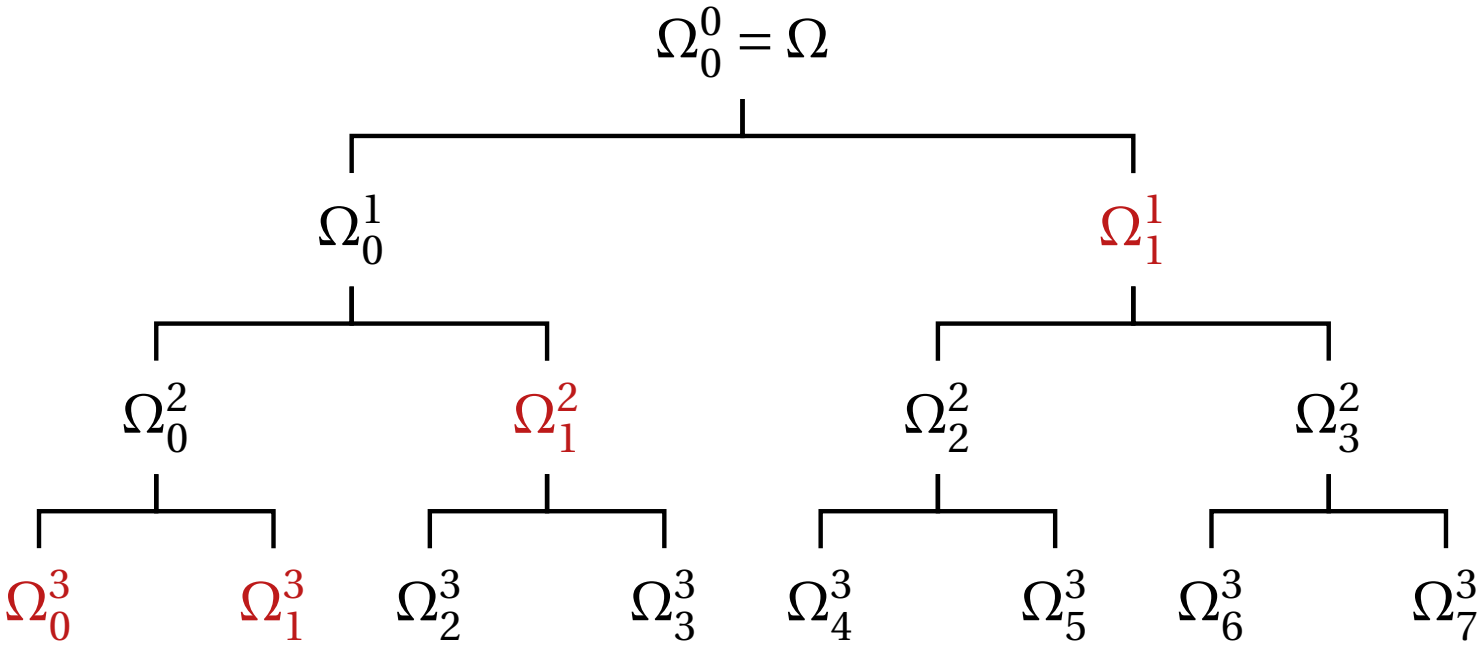


Figure: A binary partition tree of the input space Ω

- 1D Hierarchical BDCT $\Longleftrightarrow \Omega = [0,1]$ on the spatial axis
- HGLET, GHWT, eGHWT $\Longleftrightarrow \Omega = G$, a graph
- Classical wavelet packets $\Longleftrightarrow \Omega = [0, \frac{1}{2}]$ on the *frequency* axis
- *Smooth graph wavelet packets* $\Longleftrightarrow \Omega =$??

Problems of Interpreting "eigenvalues ≈ frequencies^2"

- Using graph Laplacian eigenvectors as "cosines" or Fourier modes on graphs with eigenvalues as (the square of) their "frequencies" has been quite popular.
- However, the notion of *frequency* is ill-defined on general graphs; Fourier modes on graphs may be quite different from those on regular lattices.
- Graph Laplacian eigenvectors may also exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!
- Spectral Graph Wavelet Transform (SGWT) of Hammond et al. derived wavelets on a graph based on *the Littlewood-Paley theory* that organized the graph Laplacian eigenvectors corresponding to *dyadic partitions of eigenvalues* by viewing the eigenvalues as "frequencies"
- Unfortunately, this view may face difficulty for graphs more complicated than very simple undirected unweighted paths and cycles.

A Simple Yet Important Example: A Path Graph

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

The eigenvectors of this matrix are exactly the *DCT Type II* basis vectors (used for the JPEG standard) while those of the *symmetrically-normalized Graph Laplacian matrix* $L_{\mathrm{sym}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$ are the *DCT Type I* basis! (See G. Strang, "The discrete cosine transform," *SIAM Review*, vol. 41, pp. 135–147, 1999).

- $\lambda_k = 2 - 2\cos(\pi k/n) = 4\sin^2(\pi k/2n)$, $k = 0 : n-1$.
- $\boldsymbol{\phi}_k(\ell) = a_{k;n} \cos\left(\pi k \left(\ell + \frac{1}{2}\right)/n\right)$, $k, \ell = 0 : n-1$; $a_{k;n}$ is a const. s.t. $\|\boldsymbol{\phi}_k\|_2 = 1$.
- In this simple case, λ (eigenvalue) is a monotonic function w.r.t. the frequency, which is the eigenvalue index k. *For a general graph, however, the notion of frequency is not well defined.*

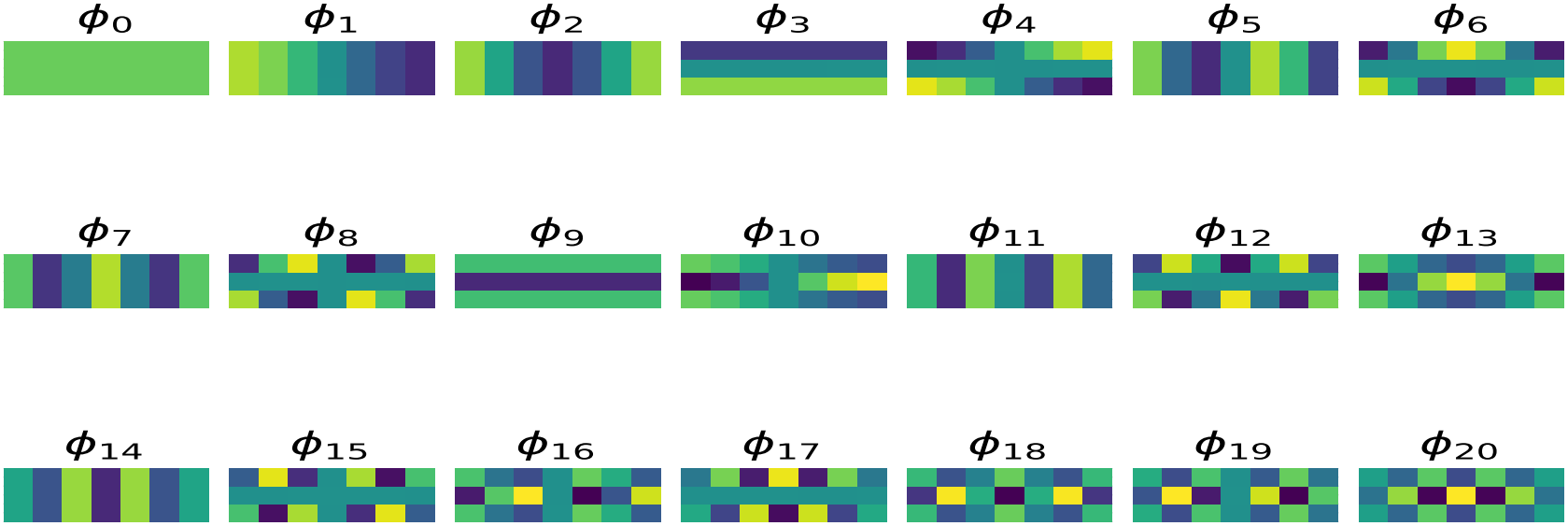
Problem with 2D Lattice Graph

- As soon as the domain becomes *even slightly more complicated than* unweighted and undirected paths/cycles, the situation completely changes: *we cannot view the eigenvalues as a simple monotonic function of frequency anymore.*
- For example, consider a thin strip in $\mathbb{R}^2$, and suppose that the domain is discretized as $P_m \times P_n$ $(m > n)$, whose Laplacian eigenpairs are:

$$\lambda_k = 4\left[\sin^2\left(\frac{\pi k_x}{2m}\right) + \sin^2\left(\frac{\pi k_y}{2n}\right)\right],$$

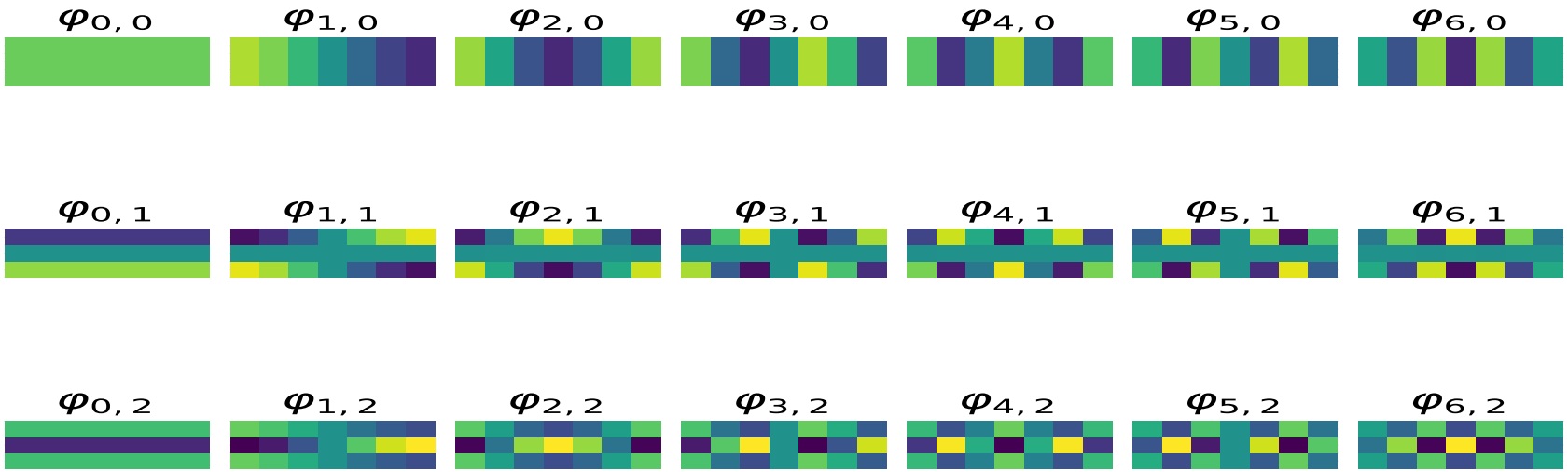
$$\phi_k(x,y) = a_{k_x;m}\, a_{k_y;n} \cos\left(\frac{\pi k_x}{m}\left(x+\frac{1}{2}\right)\right)\cos\left(\frac{\pi k_y}{n}\left(y+\frac{1}{2}\right)\right),$$

  where $k = 0:mn-1$; $k_x = 0:m-1$; $k_y = 0:n-1$; $x = 0:m-1$; and $y = 0:n-1$.
- As always, let $\{\lambda_k\}_{k=0:mn-1}$ be ordered in the nondecreasing manner. In this case, the smallest eigenvalue is still $\lambda_0 = \lambda_{(0,0)} = 0$, and the corresponding eigenvector is constant.

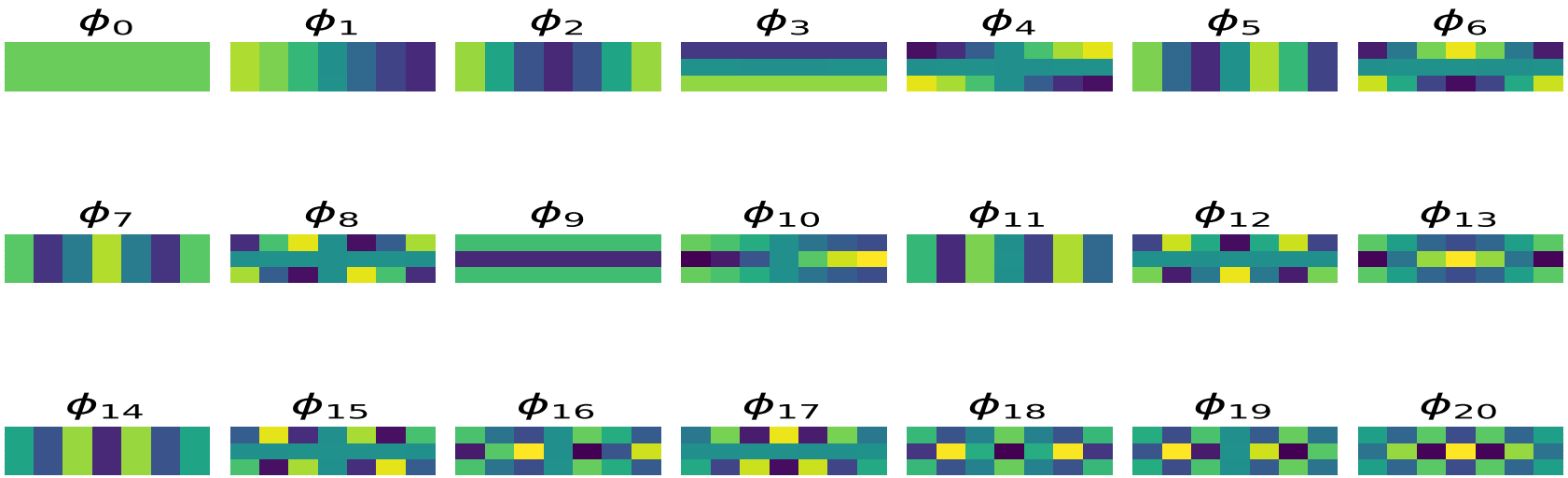


- All of a sudden the eigenvalue of a completely different type of oscillation sneaks into the eigenvalue sequence.
- Hence, on a general domain or a general graph, by simply looking at the Laplacian eigenvalue sequence $\{\lambda_k\}_{k=0,1,\ldots}$, it is *almost impossible to organize the eigenpairs into physically meaningful dyadic blocks* and apply the Littlewood-Paley approach unless the underlying domain is of very simple nature, e.g., P_n or C_n.
- For complicated domains, the notion of *frequency* is not well-defined anymore.

What we want to do is to *organize* those eigenvectors as



instead of



Outline

Plan

- How can we *quantify the difference between the eigenvectors*?
- The usual ℓ^2-distance doesn't work since $\left\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\right\|_2 = \sqrt{2}\delta_{i\neq j}$.
- Need to come up with a metric that quantifies the "behavioral" differences between any pair of eigenvectors. Having such a metric, we do the following:

1. Choose a metric and compute the "distance" between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$ for all $i, j = 0 : n-1$, which results in a "distance" matrix $D \in \mathbb{R}^{n\times n}_{\geq 0}$
2. Construct a *dual graph* $G^\star(V^\star, E^\star)$ where the ith node corresponds to $\boldsymbol{\phi}_i$, and the weight of the edge (i, j) is the *affinity* between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$, e.g., $1/D_{i,j}$ or $\exp(-D^2_{i,j}/\epsilon)$ with some $\epsilon > 0$
3. *Organize* and *group* $V^\star$ to generate *wavelet packet vectors* on G

Plan

- How can we *quantify the difference between the eigenvectors*?
- The usual ℓ^2-distance doesn't work since $\left\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\right\|_2 = \sqrt{2}\delta_{i \neq j}$.
- Need to come up with a metric that quantifies the "behavioral" differences between any pair of eigenvectors. Having such a metric, we do the following:

1. Choose a metric and compute the "distance" between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$ for all $i, j = 0 : n-1$, which results in a "distance" matrix $D \in \mathbb{R}_{\geq 0}^{n \times n}$
2. Construct a *dual graph* $G^\star(V^\star, E^\star)$ where the ith node corresponds to $\boldsymbol{\phi}_i$, and the weight of the edge (i, j) is the *affinity* between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$, e.g., $1/D_{i,j}$ or $\exp(-D_{i,j}^2/\epsilon)$ with some $\epsilon > 0$
3. *Organize* and *group* $V^\star$ to generate *wavelet packet vectors* on G

Plan

- How can we *quantify the difference between the eigenvectors*?
- The usual ℓ^2-distance doesn't work since $\left\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\right\|_2 = \sqrt{2}\delta_{i\neq j}$.
- Need to come up with a metric that quantifies the "behavioral" differences between any pair of eigenvectors. Having such a metric, we do the following:

1. Choose a metric and compute the "distance" between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$ for all $i, j = 0 : n-1$, which results in a "distance" matrix $D \in \mathbb{R}^{n\times n}_{\geq 0}$
2. Construct a *dual graph* $G^\star(V^\star, E^\star)$ where the ith node corresponds to $\boldsymbol{\phi}_i$, and the weight of the edge (i, j) is the *affinity* between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$, e.g., $1/D_{i,j}$ or $\exp(-D^2_{i,j}/\epsilon)$ with some $\epsilon > 0$
3. *Organize* and *group* $V^\star$ to generate *wavelet packet vectors* on G

Plan

- How can we *quantify the difference between the eigenvectors*?
- The usual ℓ^2-distance doesn't work since $\left\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\right\|_2 = \sqrt{2}\delta_{i\neq j}$.
- Need to come up with a metric that quantifies the "behavioral" differences between any pair of eigenvectors. Having such a metric, we do the following:

1. Choose a metric and compute the "distance" between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$ for all $i, j = 0 : n-1$, which results in a "distance" matrix $D \in \mathbb{R}^{n\times n}_{\geq 0}$
2. Construct a *dual graph* $G^\star(V^\star, E^\star)$ where the ith node corresponds to $\boldsymbol{\phi}_i$, and the weight of the edge (i, j) is the *affinity* between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$, e.g., $1/D_{i,j}$ or $\exp(-D^2_{i,j}/\epsilon)$ with some $\epsilon > 0$
3. *Organize* and *group* $V^\star$ to generate *wavelet packet vectors* on G

Plan

- How can we *quantify the difference between the eigenvectors*?
- The usual ℓ^2-distance doesn't work since $\left\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\right\|_2 = \sqrt{2}\delta_{i\neq j}$.
- Need to come up with a metric that quantifies the "behavioral" differences between any pair of eigenvectors. Having such a metric, we do the following:

1. Choose a metric and compute the "distance" between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$ for all $i, j = 0 : n-1$, which results in a "distance" matrix $D \in \mathbb{R}^{n\times n}_{\geq 0}$
2. Construct a *dual graph* $G^\star(V^\star, E^\star)$ where the ith node corresponds to $\boldsymbol{\phi}_i$, and the weight of the edge (i, j) is the *affinity* between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$, e.g., $1/D_{i,j}$ or $\exp(-D^2_{i,j}/\epsilon)$ with some $\epsilon > 0$
3. *Organize* and *group* $V^\star$ to generate *wavelet packet vectors* on G

Plan

- How can we *quantify the difference between the eigenvectors*?
- The usual ℓ^2-distance doesn't work since $\left\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\right\|_2 = \sqrt{2}\delta_{i\neq j}$.
- Need to come up with a metric that quantifies the "behavioral" differences between any pair of eigenvectors. Having such a metric, we do the following:

1. Choose a metric and compute the "distance" between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$ for all $i, j = 0 : n-1$, which results in a "distance" matrix $D \in \mathbb{R}^{n\times n}_{\geq 0}$
2. Construct a *dual graph* $G^\star(V^\star, E^\star)$ where the ith node corresponds to $\boldsymbol{\phi}_i$, and the weight of the edge (i, j) is the *affinity* between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$, e.g., $1/D_{i,j}$ or $\exp(-D^2_{i,j}/\epsilon)$ with some $\epsilon > 0$
3. *Organize* and *group* $V^\star$ to generate *wavelet packet vectors* on G

Various Metrics for Eigenvector Differences

- A *similarity* measure (HAD) based on the *average of local correlations* of eigenvectors [A. Cloninger & S. Steinerberger, 2018]
- The *ramified optimal transport* (ROT) method [N. Saito, 2018]
- The *difference of absolute gradient* (DAG) method [H. Li & N. Saito, 2019]
- The *time-stepping diffusion* (TSD) method [H. Li & N. Saito, 2019]
- The ROT method seems to work well for transportation networks with hubs (e.g., neuronal dendritic trees) whereas the HAD and DAG methods seem to work well for (ir)regular grids/lattices (e.g., road networks).
- For more pros and cons of these methods, see [H. Li & N. Saito, 2019].

Difference of Absolute Gradient (DAG) Pseudometric

- The basic idea: use the *absolute gradient* of each eigenvector as its feature vector describing its behavior.
- Let $G(V,E)$ be an input graph (connected, undirected, weighted, and simple) with $|V|=n$, $|E|=m$. Let $Q \in \mathbb{R}^{n\times m}$ be its *incidence matrix*. Then, *DAG **pseudo**metric* (*the identity of discernible* is not satisfied) is defined as:

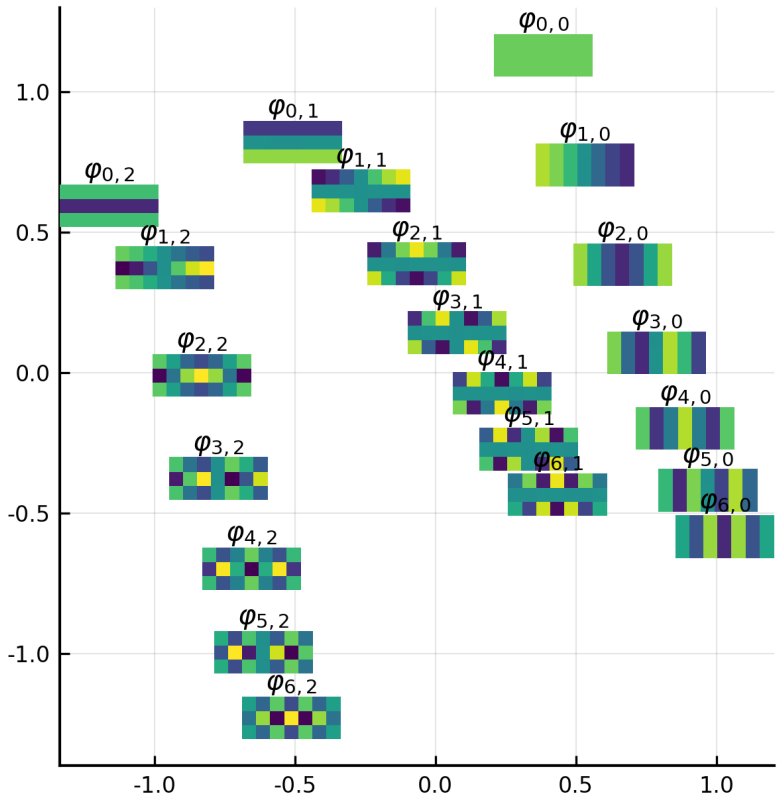
$$d_{\mathrm{DAG}}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_j) := \||\nabla_G|\boldsymbol{\phi}_i - |\nabla_G|\boldsymbol{\phi}_j\|_2 \quad \text{where } |\nabla_G|\boldsymbol{\phi} := \mathrm{abs.}(Q^{\mathsf{T}}\boldsymbol{\phi}) \in \mathbb{R}^m_{\geq 0}$$

- It is related to the *Hadamard product-based affinity* proposed by [A. Cloninger & S. Steinerberger, 2018] as

$$\begin{aligned} d_{\mathrm{DAG}}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_j)^2 &= \left\langle |\nabla_G|\boldsymbol{\phi}_i - |\nabla_G|\boldsymbol{\phi}_j, |\nabla_G|\boldsymbol{\phi}_i - |\nabla_G|\boldsymbol{\phi}_j \right\rangle_E \\ &= \left\langle |\nabla_G|\boldsymbol{\phi}_i, |\nabla_G|\boldsymbol{\phi}_i \right\rangle_E + \left\langle |\nabla_G|\boldsymbol{\phi}_j, |\nabla_G|\boldsymbol{\phi}_j \right\rangle_E - 2\left\langle |\nabla_G|\boldsymbol{\phi}_i, |\nabla_G|\boldsymbol{\phi}_j \right\rangle_E \\ &= \lambda_i + \lambda_j - \sum_{x\in V}\sum_{y\sim x} |\boldsymbol{\phi}_i(x) - \boldsymbol{\phi}_i(y)| \cdot |\boldsymbol{\phi}_j(x) - \boldsymbol{\phi}_j(y)| \quad \text{thanks to } QQ^{\mathsf{T}} = L \end{aligned}$$

  where $\langle\cdot,\cdot\rangle_E$ is the inner product over edges.
- The last term of the formula can be viewed as *a global average of absolute local correlation* between eigenvectors $\Longrightarrow$ the Hadamard-product affinity.
- Given the eigenvectors, *the computational cost is $O(m)$* for each $d_{\mathrm{DAG}}(\cdot,\cdot)$ eval.

2D Lattice $P_7 \times P_3$: d_{DAG} visualized by the classical MDS



- d_{DAG} nicely detects two directions of the oscillations and the eigenvectors are organized naturally in the *2D lattice*.
- For each column of the lattice, the eigenvectors have the same oscillation pattern in the y-direction and the oscillation in the x-direction increases linearly, and vice versa.

Outline

Natural Graph Wavelet Packets: Basic Steps

1. *Bipartition the dual graph $G^\star$ recursively* via any method, e.g., spectral graph bipartition using the *Fiedler vectors*
2. Generate wavelet packet vectors on each subgraph of $G^\star$ that are *well localized on G*

- We refer to the graph wavelet packets generated by the above strategy as *Natural Graph Wavelet Packets* (NGWPs).
- $\exists$ Several possibilities in Step 2; will discuss only one of them today.
- Let $\Phi_0^{(0)} := [\boldsymbol{\phi}_0, \ldots, \boldsymbol{\phi}_{n-1}]$ = a matrix representation of the GL eigenvectors of G = the node set $V^\star$ of $G^\star$, and suppose we get the following *hierarchical bipartition tree* of $\Phi_0^{(0)}$:

Natural Graph Wavelet Packets: Basic Steps

1. *Bipartition the dual graph $G^\star$ recursively* via any method, e.g., spectral graph bipartition using the *Fiedler vectors*
2. Generate wavelet packet vectors on each subgraph of $G^\star$ that are *well localized on G*

- We refer to the graph wavelet packets generated by the above strategy as *Natural Graph Wavelet Packets* (NGWPs).
- $\exists$ Several possibilities in Step 2; will discuss only one of them today.
- Let $\Phi_0^{(0)} := [\boldsymbol{\phi}_0, \ldots, \boldsymbol{\phi}_{n-1}]$ = a matrix representation of the GL eigenvectors of G = the node set $V^\star$ of $G^\star$, and suppose we get the following *hierarchical bipartition tree* of $\Phi_0^{(0)}$:

Natural Graph Wavelet Packets: Basic Steps

1. *Bipartition the dual graph $G^\star$ recursively* via any method, e.g., spectral graph bipartition using the *Fiedler vectors*
2. Generate wavelet packet vectors on each subgraph of $G^\star$ that are *well localized on G*

- We refer to the graph wavelet packets generated by the above strategy as *Natural Graph Wavelet Packets* (NGWPs).
- $\exists$ Several possibilities in Step 2; will discuss only one of them today.
- Let $\Phi_0^{(0)} := [\boldsymbol{\phi}_0, \ldots, \boldsymbol{\phi}_{n-1}]$ = a matrix representation of the GL eigenvectors of G = the node set $V^\star$ of $G^\star$, and suppose we get the following *hierarchical bipartition tree* of $\Phi_0^{(0)}$:

Natural Graph Wavelet Packets: Basic Steps

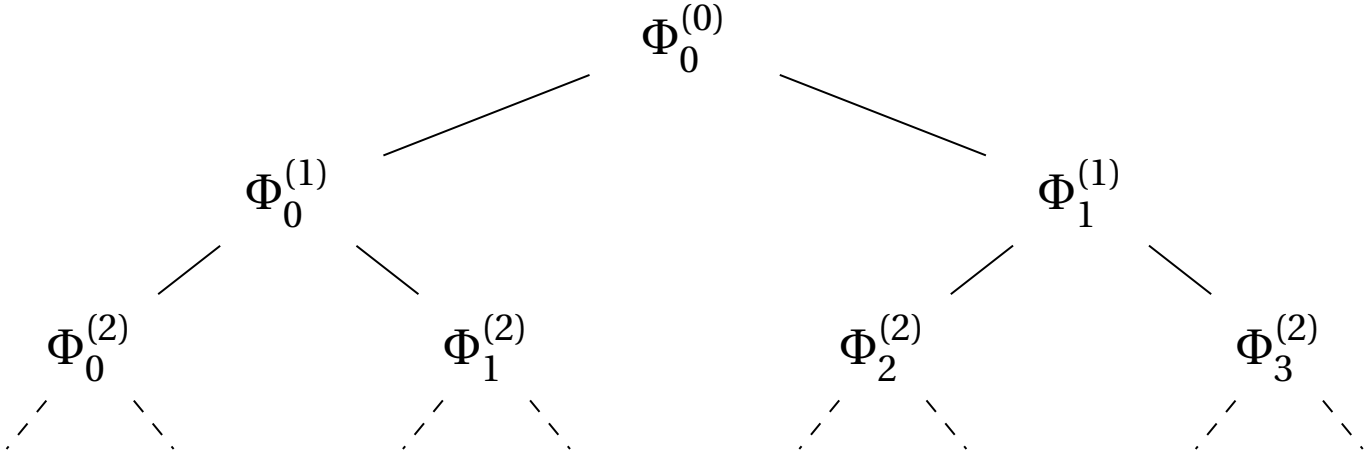
1. *Bipartition the dual graph $G^\star$ recursively* via any method, e.g., spectral graph bipartition using the *Fiedler vectors*
2. Generate wavelet packet vectors on each subgraph of $G^\star$ that are *well localized on G*

- We refer to the graph wavelet packets generated by the above strategy as *Natural Graph Wavelet Packets* (NGWPs).
- $\exists$ Several possibilities in Step 2; will discuss only one of them today.
- Let $\Phi_0^{(0)} := [\boldsymbol{\phi}_0, \ldots, \boldsymbol{\phi}_{n-1}]$ = a matrix representation of the GL eigenvectors of G = the node set $V^\star$ of $G^\star$, and suppose we get the following *hierarchical bipartition tree* of $\Phi_0^{(0)}$:

Natural Graph Wavelet Packets: Basic Steps

1. *Bipartition the dual graph $G^\star$ recursively* via any method, e.g., spectral graph bipartition using the *Fiedler vectors*
2. Generate wavelet packet vectors on each subgraph of $G^\star$ that are *well localized on G*

- We refer to the graph wavelet packets generated by the above strategy as *Natural Graph Wavelet Packets* (NGWPs).
- $\exists$ Several possibilities in Step 2; will discuss only one of them today.
- Let $\Phi_0^{(0)} := [\boldsymbol{\phi}_0, \ldots, \boldsymbol{\phi}_{n-1}]$ = a matrix representation of the GL eigenvectors of G = the node set $V^\star$ of $G^\star$, and suppose we get the following *hierarchical bipartition tree* of $\Phi_0^{(0)}$:



NGWPs by Varimax Rotation

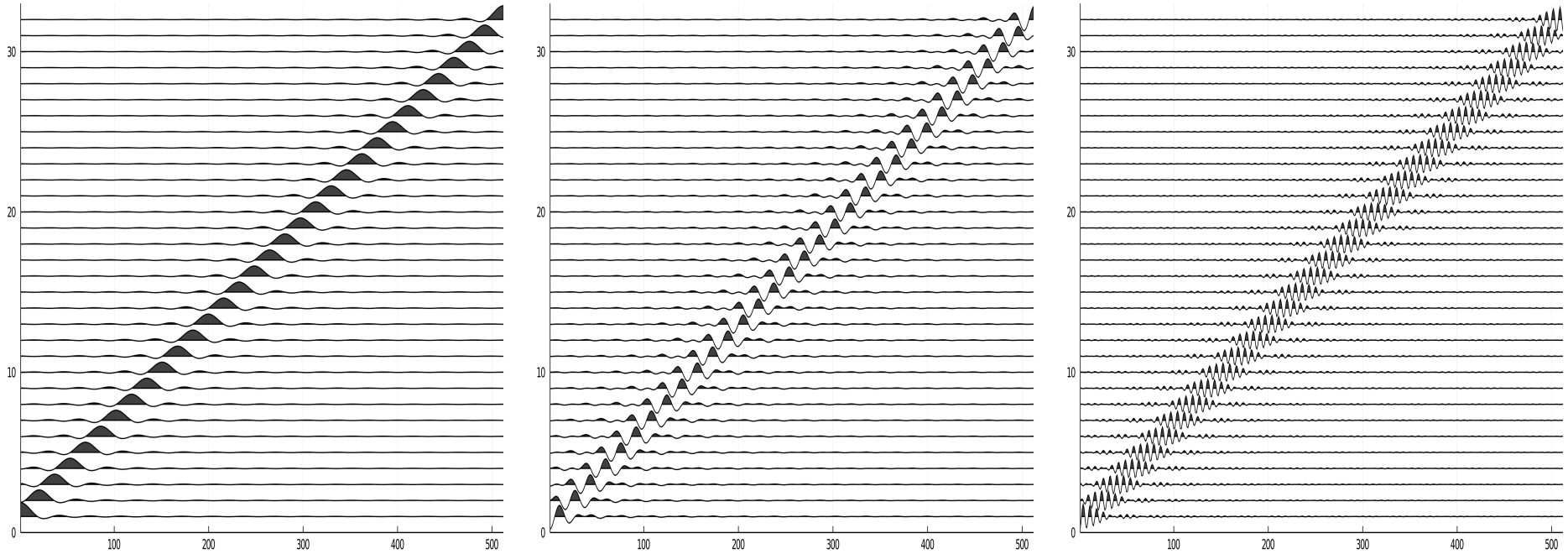
- Given the full binary partition tree of $\Phi_0^{(0)} \in \mathbb{R}^{n\times n}$, perform the *varimax rotation* on $\Phi_k^{(j)} \in \mathbb{R}^{n\times n_k^j}$ for each j and k.
- Varimax rotation [Kaiser (1958); Jennrich (2001)] is an orthogonal rotation, often used in *factor analysis*, to maximize the variances of energy distribution (or a scaled version of the *kurtosis*) of the input column vectors.
- Thanks to the orthonormality of columns of $\Phi_k^{(j)}$, this is equivalent to finding an orthogonal rotation that maximizes the overall *4th order moments*, i.e.,

$$\Psi_k^{(j)} := \Phi_k^{(j)} \cdot R_k^{(j)}, \quad \text{where } R_k^{(j)} = \arg \max_{R\in \mathrm{SO}(n_k^j)} \sum_{p=1}^{n} \sum_{q=1}^{n_k^j} \left[\left(\Phi_k^{(j)} \cdot R \right)^4 \right]_{p,q}.$$

- The column vectors of $\Psi_k^{(j)}$ are *more "localized" in the original domain* G than those of $\Phi_k^{(j)}$. This type of localization is important since the GL eigenvectors in $\Phi_k^{(j)}$ are of *global* nature in general.

The varimax-rotation NGWPs on P_n

are essentially the *Shannon wavelet packets !*



(a) Father wavelet vectors $\Psi_0^{(4)}$

(b) Mother wavelet vectors $\Psi_1^{(4)}$

(c) Wavelet packet vectors $\Psi_4^{(4)}$

Figure: Some of the Shannon wavelet packet vectors on P_{512}

Selecting a Good Natural Graph Wavelet Packet Basis

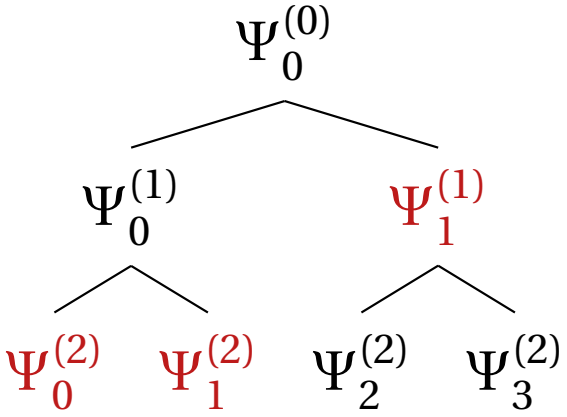
- Once the NGWP dictionary is built, one can apply the *best-basis selection algorithm* of Coifman-Wickerhauser or its variants developed by the Saito group to choose the most suitable basis for a given task (e.g., efficient approximation, denoising, classification, regression, etc.). Note that the best-basis algorithm searches the best one among more than $(1.5)^n$ possible ONBs from the wavelet packet dictionary.
- For the examples today, we used the *ℓ^1-norm minimization* to select the best (or *sparsest*) basis among the NGWP dictionary.
- Of course, if one prefers, one can also choose the *wavelet basis* from the NGWP dictionary by selecting the specific subspaces and basis vectors.

Selecting a Good Natural Graph Wavelet Packet Basis

- Once the NGWP dictionary is built, one can apply the *best-basis selection algorithm* of Coifman-Wickerhauser or its variants developed by the Saito group to choose the most suitable basis for a given task (e.g., efficient approximation, denoising, classification, regression, etc.). Note that the best-basis algorithm searches the best one among more than $(1.5)^n$ possible ONBs from the wavelet packet dictionary.
- For the examples today, we used the *ℓ^1-norm minimization* to select the best (or *sparsest*) basis among the NGWP dictionary.
- Of course, if one prefers, one can also choose the *wavelet basis* from the NGWP dictionary by selecting the specific subspaces and basis vectors.

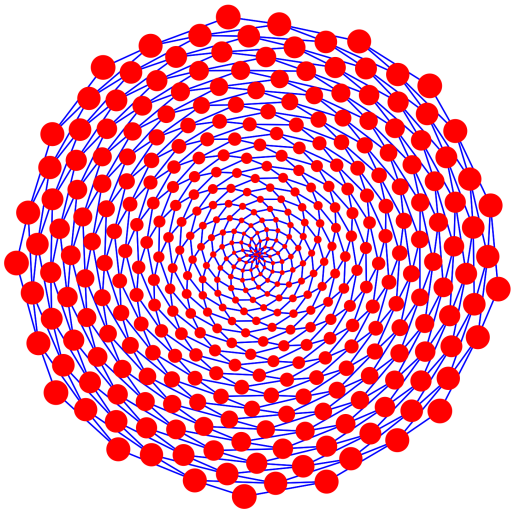
Selecting a Good Natural Graph Wavelet Packet Basis

- Once the NGWP dictionary is built, one can apply the *best-basis selection algorithm* of Coifman-Wickerhauser or its variants developed by the Saito group to choose the most suitable basis for a given task (e.g., efficient approximation, denoising, classification, regression, etc.). Note that the best-basis algorithm searches the best one among more than $(1.5)^n$ possible ONBs from the wavelet packet dictionary.
- For the examples today, we used the *ℓ^1-norm minimization* to select the best (or *sparsest*) basis among the NGWP dictionary.
- Of course, if one prefers, one can also choose the *wavelet basis* from the NGWP dictionary by selecting the specific subspaces and basis vectors.

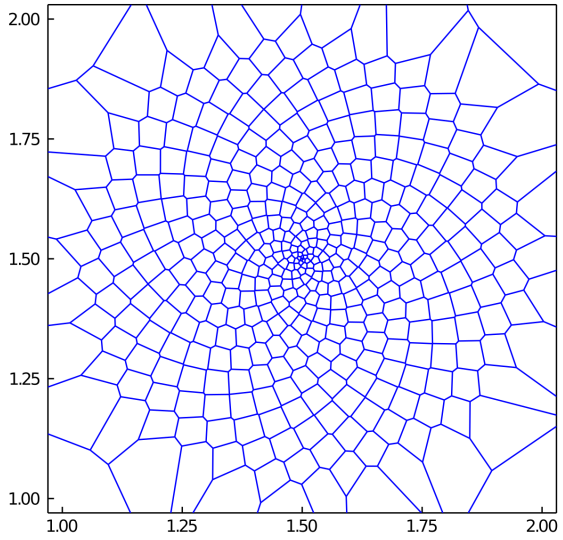


Outline

Example 1: Sunflower Graph



(a) Sunflower graph

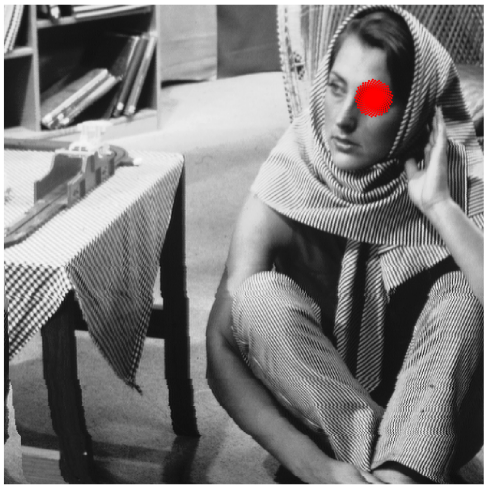


(b) Voronoi tessellation

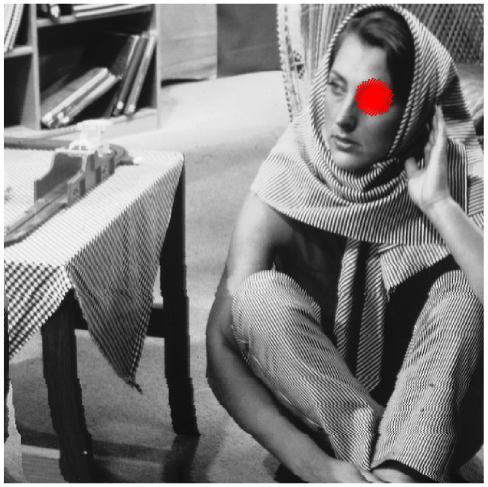
Figure: (a) Sunflower graph ($n = 400$); (b) Its Voronoi tessellation

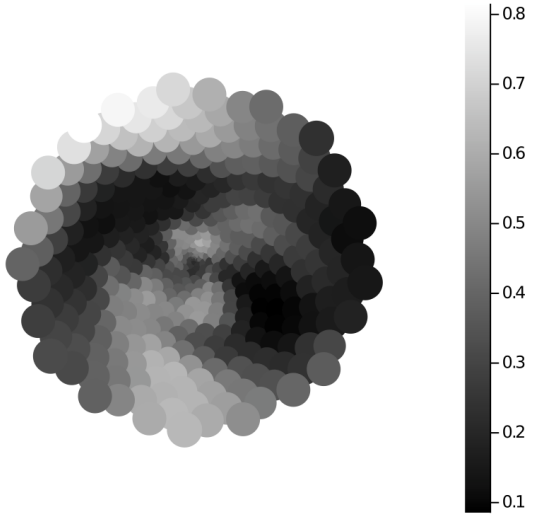
- This graph often appears in nature
- Consistent counting of spirals gives rise to *Fibonacci numbers* $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$
- Edge weights = the inverse Euclidean distances between nodes
- My view: a simple model of the distribution of *photoreceptors in mammalian visual systems* due to cell generation and growth

Example 1: Sampling an eye of Barbara by Sunflower Graph

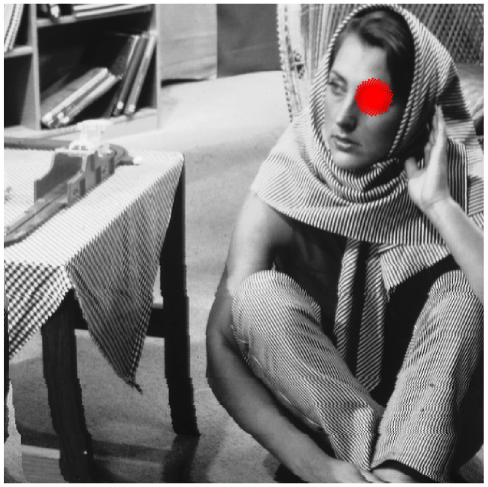


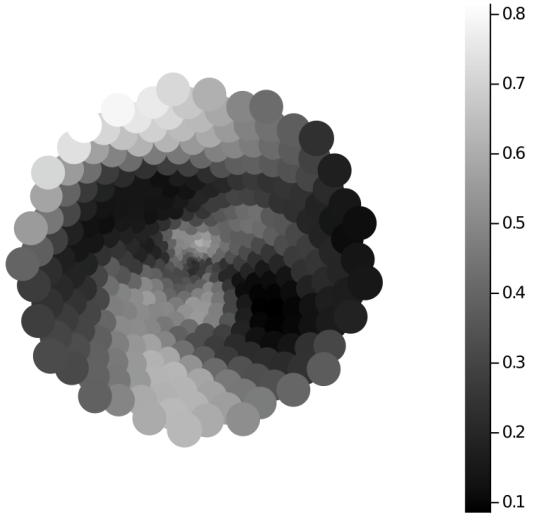
Example 1: Sampling an eye of Barbara by Sunflower Graph

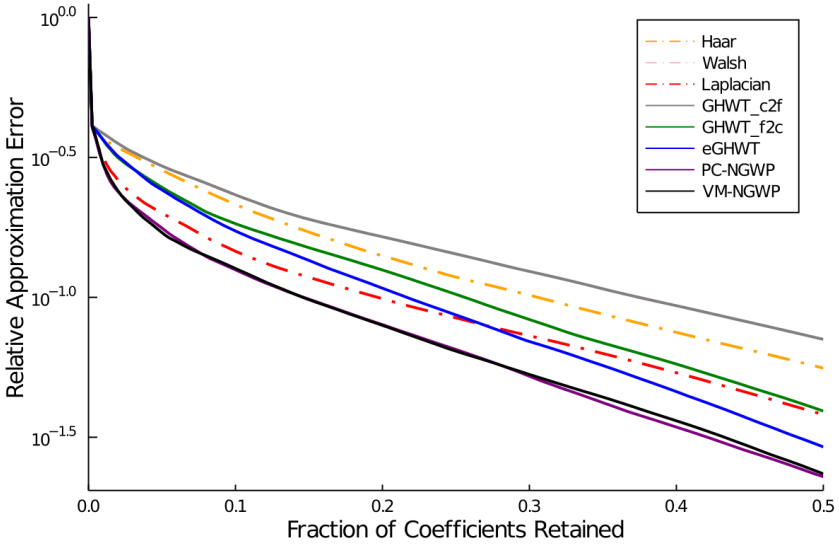




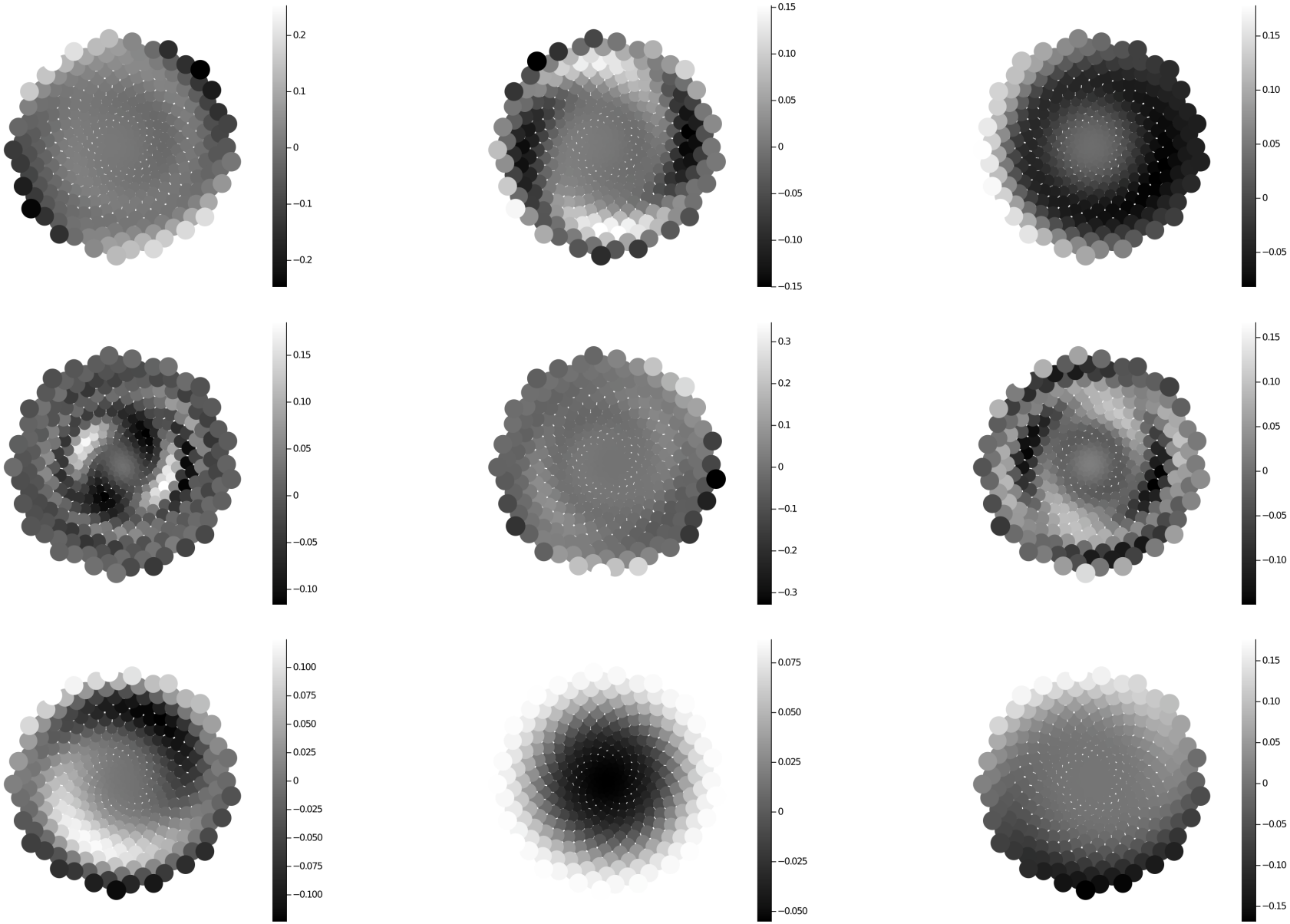
Example 1: Sampling an eye of Barbara by Sunflower Graph





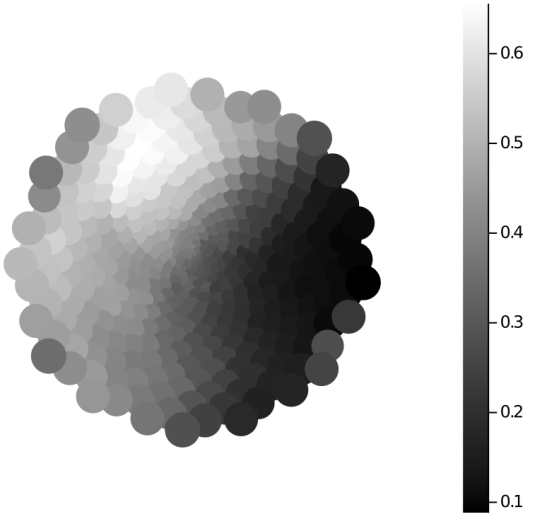


Example 1: Top 9 NGWP vectors for Barbara's eye

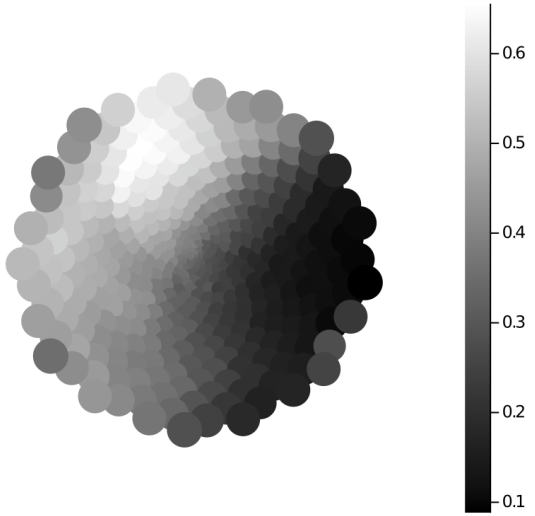


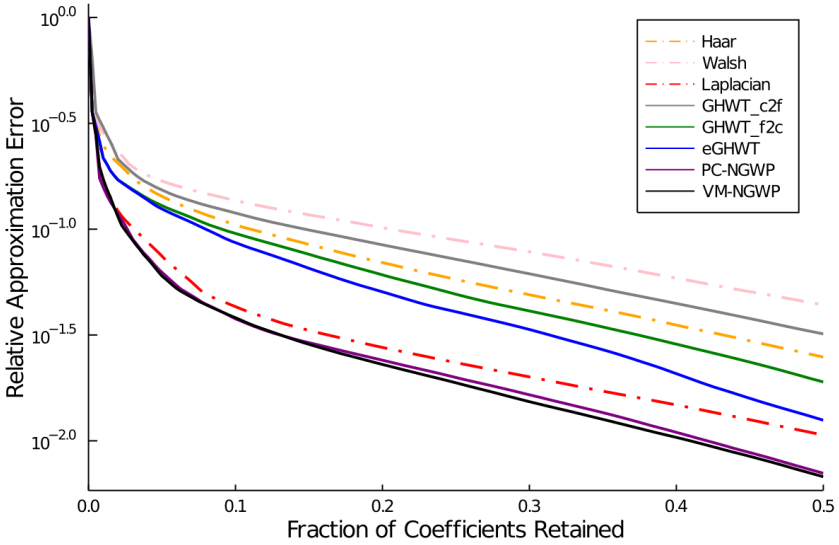
Example 1: Sampling cheek of Barbara by Sunflower Graph

Example 1: Sampling cheek of Barbara by Sunflower Graph

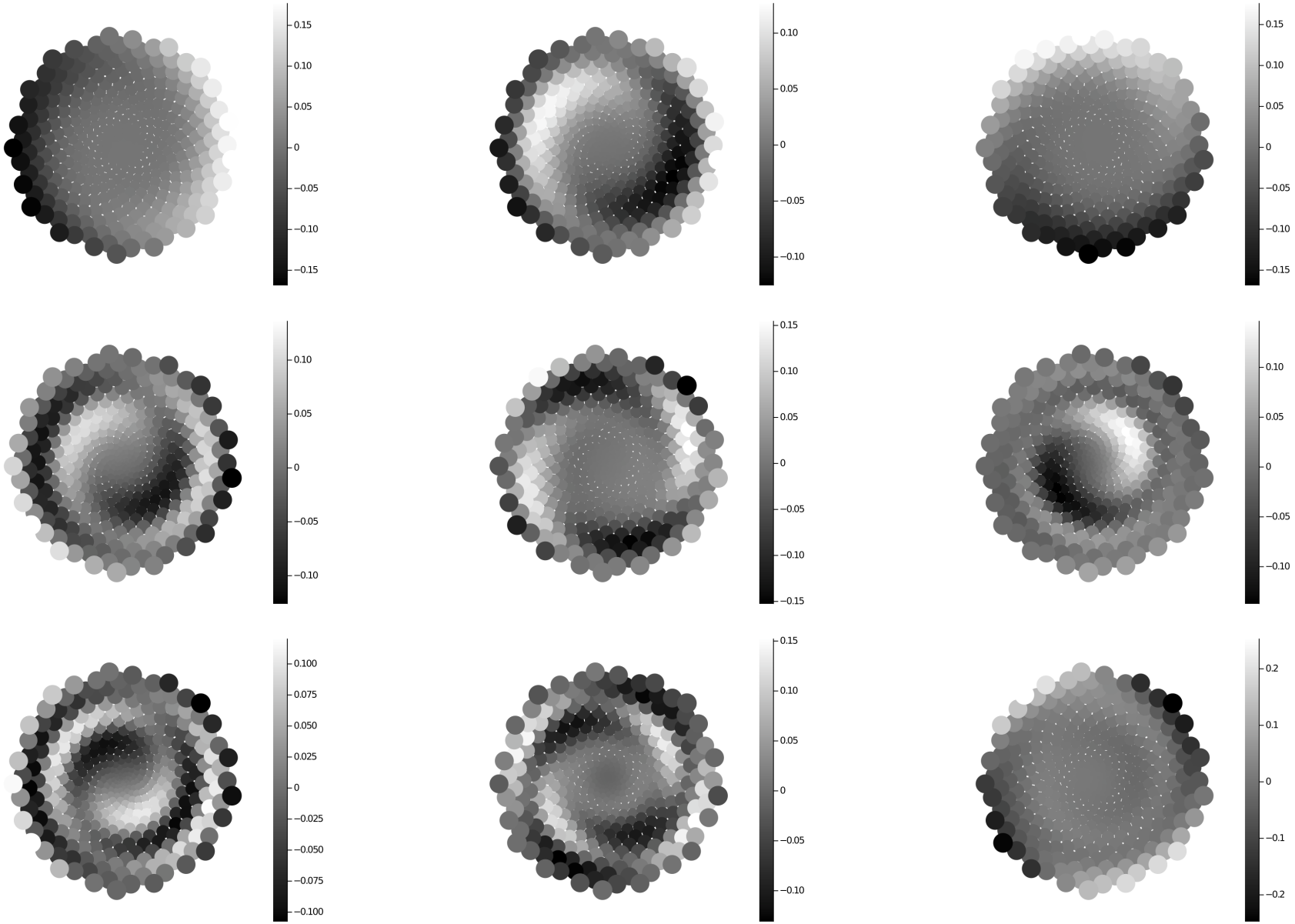


Example 1: Sampling cheek of Barbara by Sunflower Graph

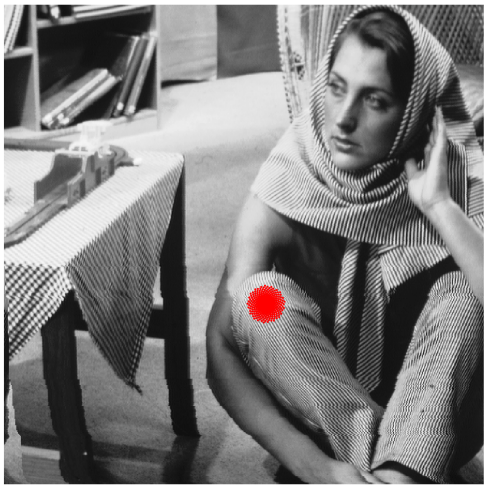




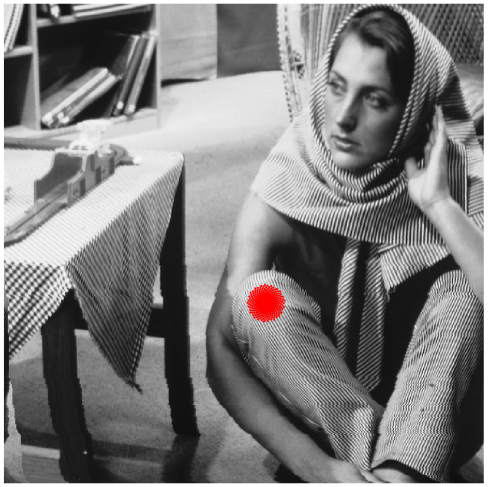
Example 1: Top 9 NGWP vectors for Barbara's cheek

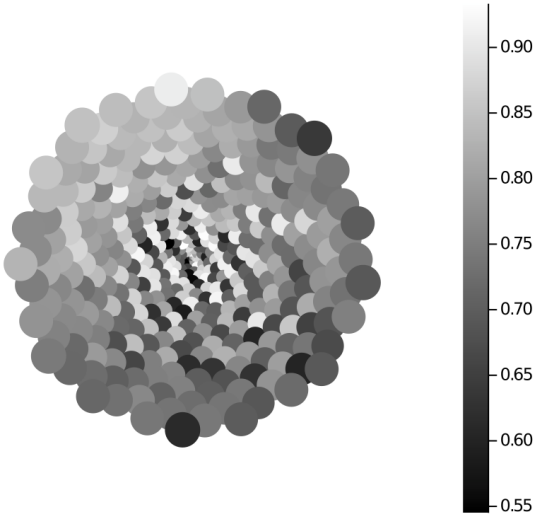


Example 1: Sampling pants of Barbara by Sunflower Graph

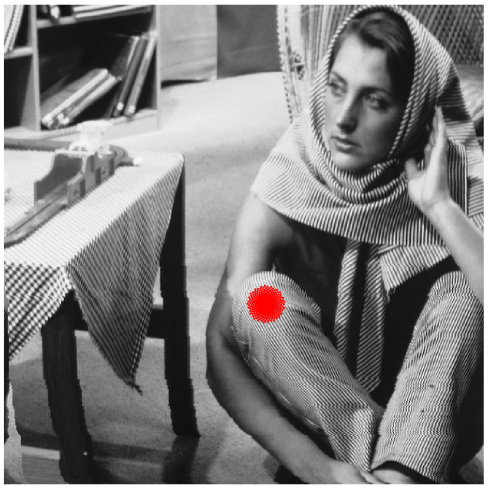


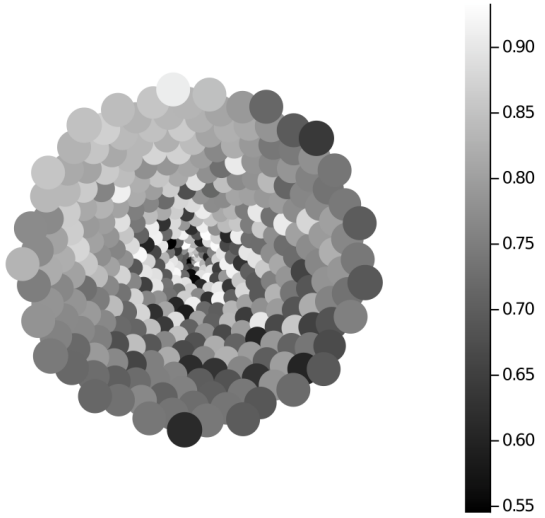
Example 1: Sampling pants of Barbara by Sunflower Graph

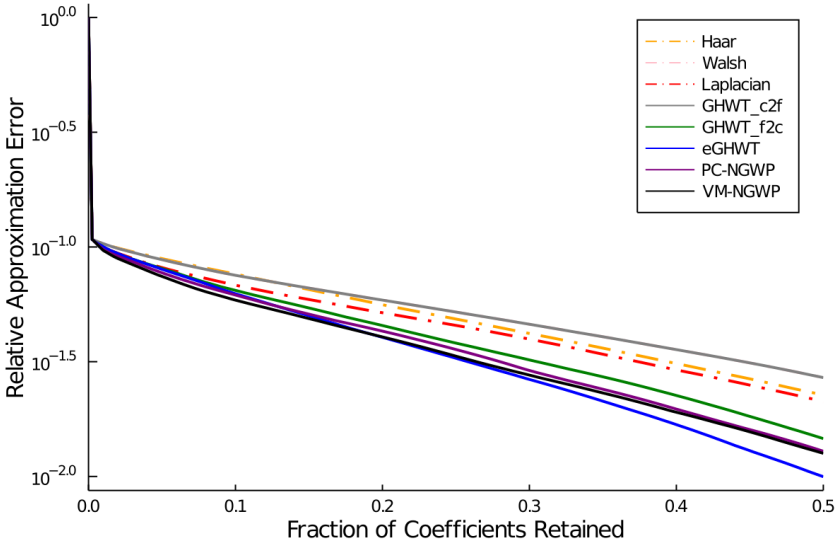




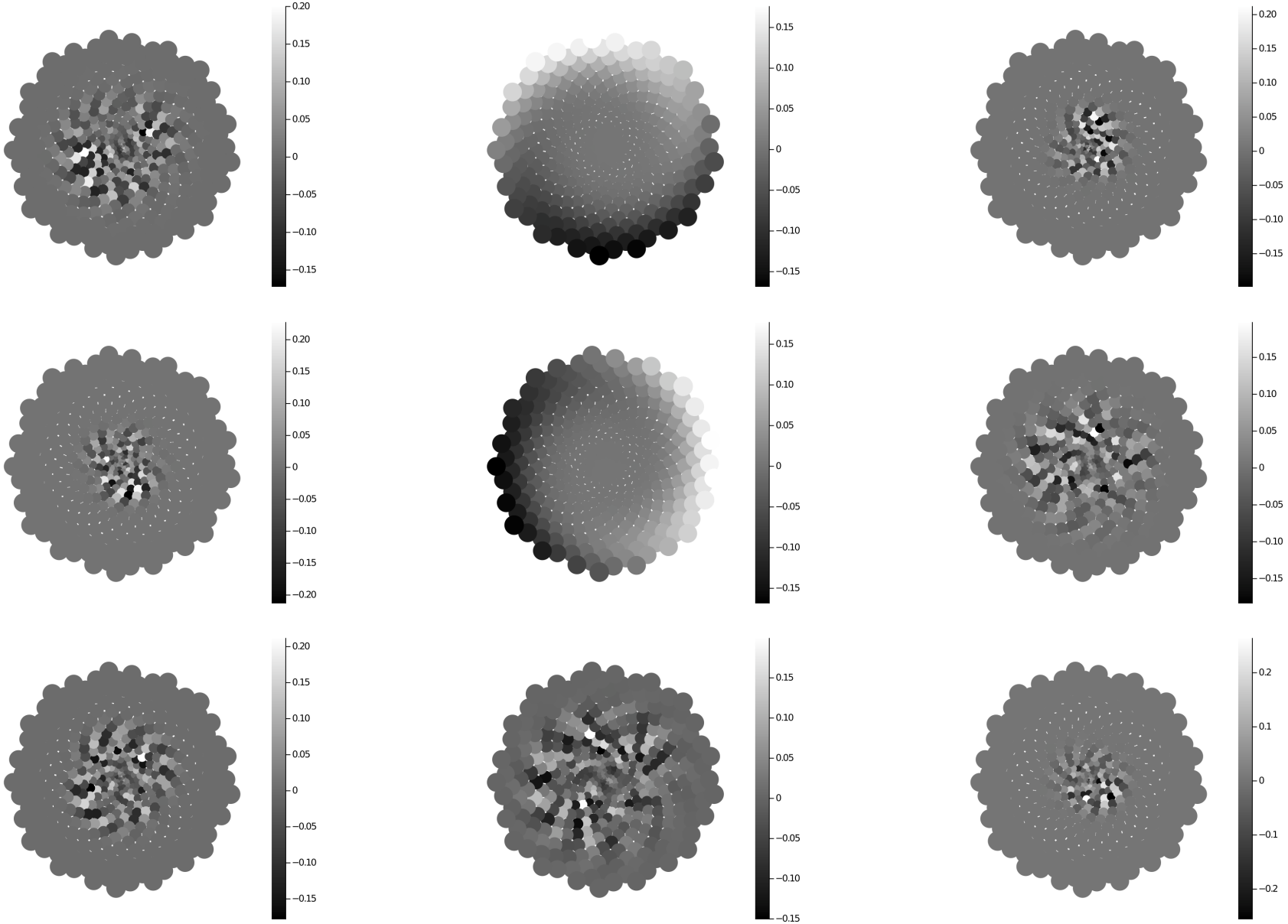
Example 1: Sampling pants of Barbara by Sunflower Graph







Example 1: Top 9 NGWP vectors for Barbara's pants



Example 2: Toronto Road Network

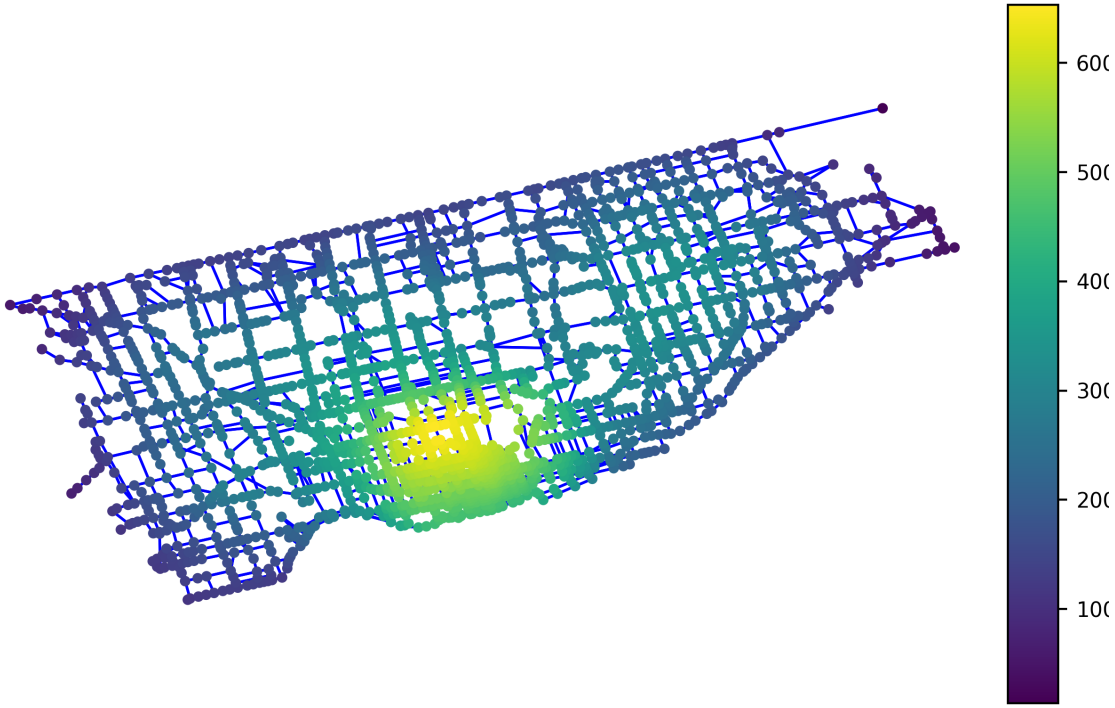
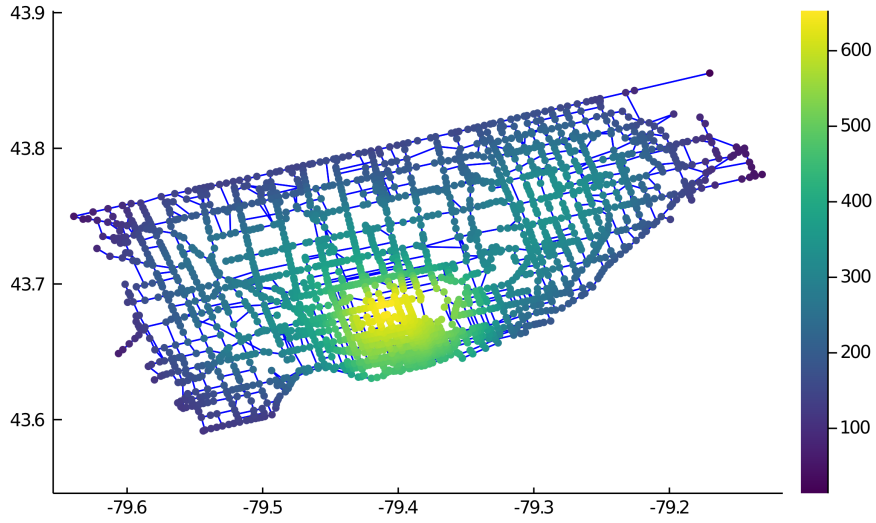


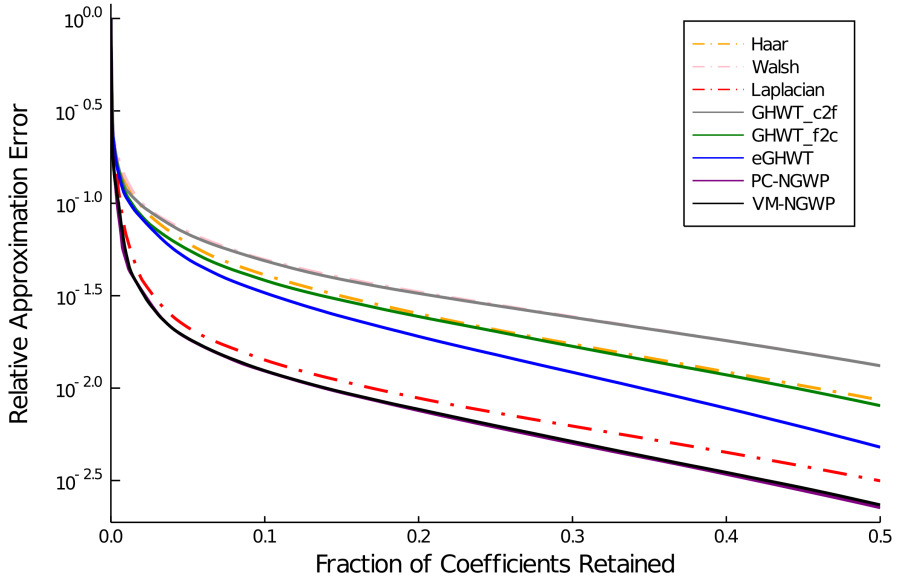
Figure: The road network of Toronto ($n = 2275$ nodes and $m = 3381$ edges)

- Nodes = intersections (with traffic lights); edges = streets
- Edge weights = the inverse Euclidean distances between nodes
- Two graph signals considered: 1) spatial distribution of the street intersections; 2) real pedestrian counts between the hours of 7:30am and 6:00pm on a single day measured during the period 03/22/2004–02/28/2018

Example 2: Smooth Histogram of Street Intersections



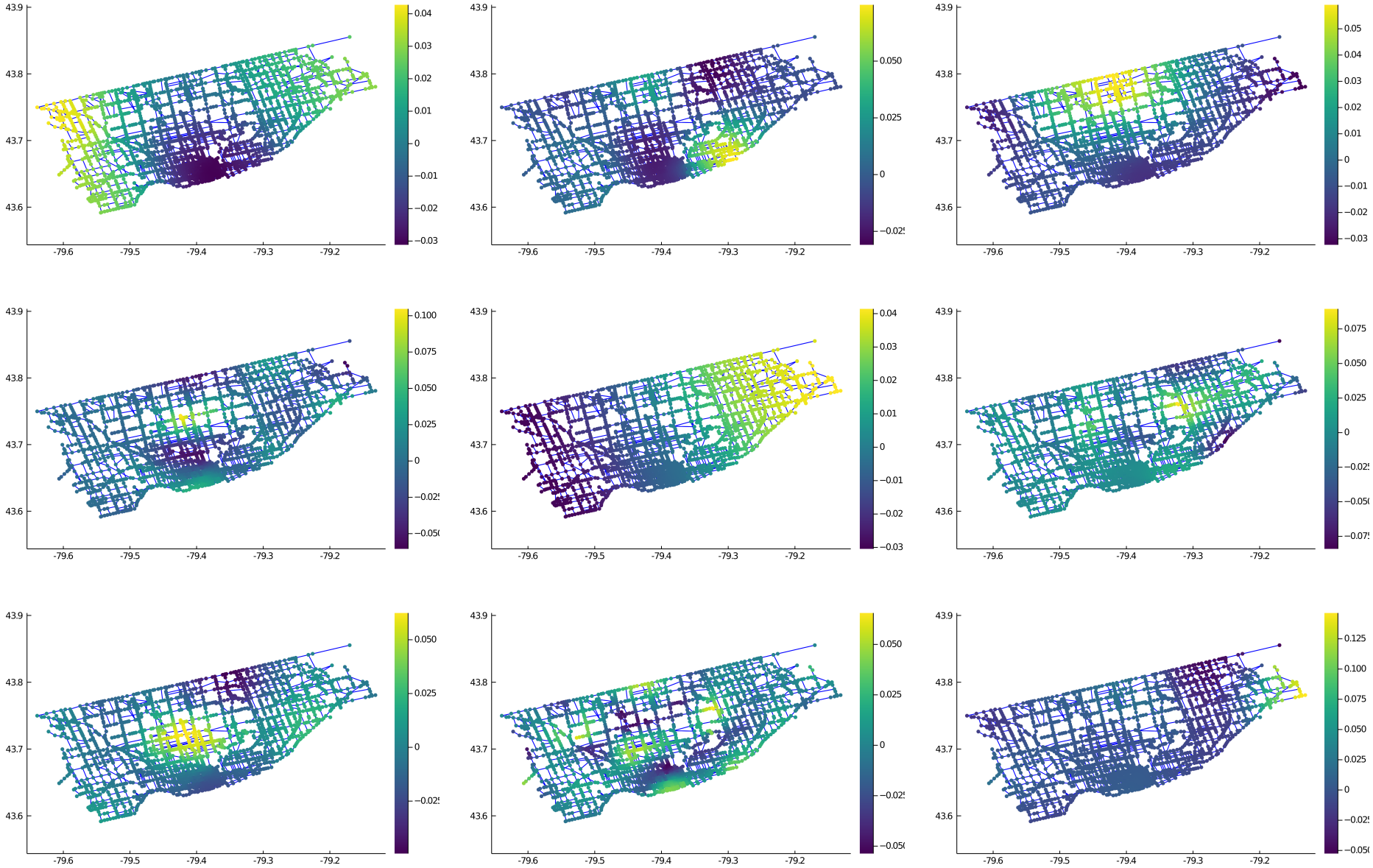
(a) Graph signal



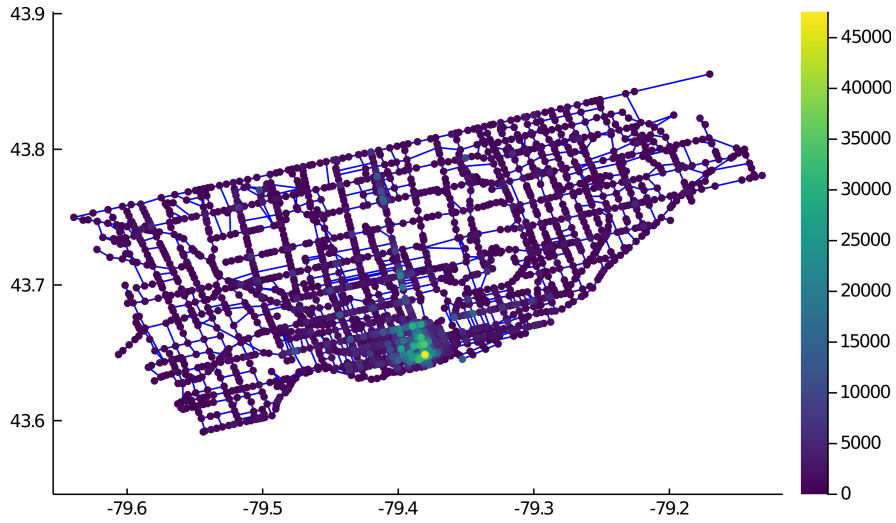
(b) Approximation

Figure: (a) A graph signal representing the smooth spatial distribution of the street intersections on the Toronto street network Toronto; (b) Relative ℓ^2 approximation error of the data shown in (a) vs the fraction of kept coefficients.

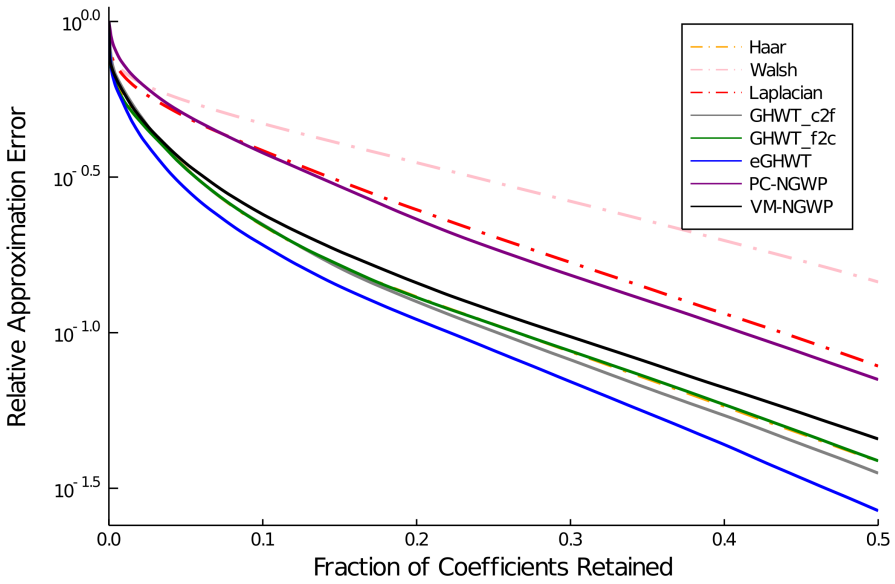
Top 9 VM-NGWP best basis vectors



Example 2: Pedestrian Counts at the Intersections



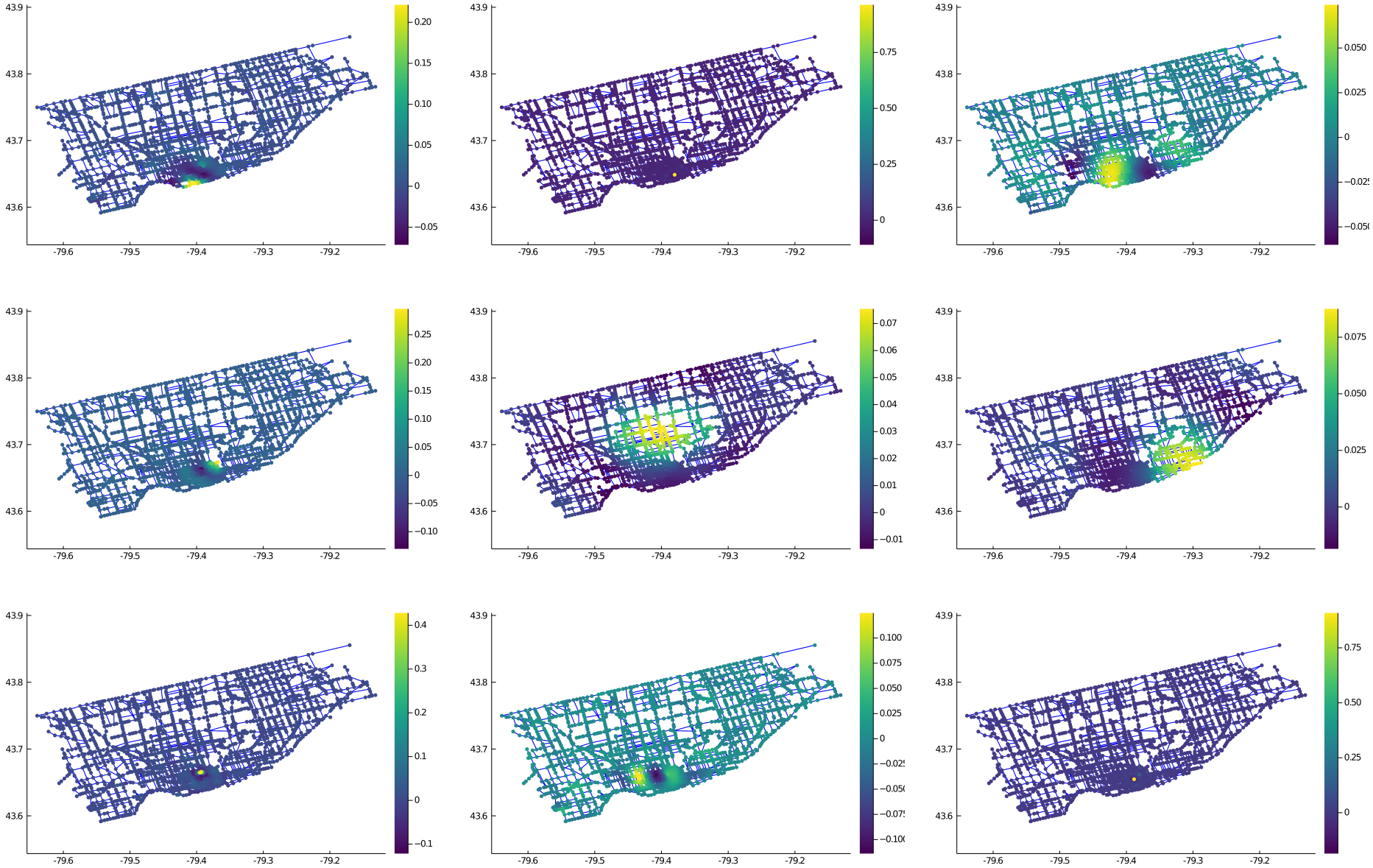
(a) Graph signal



(b) Approximation

Figure: (a) Pedestrian volume data in the city of Toronto; (b) Relative ℓ^2 approximation error of the data shown in (a) vs the fraction of kept coefficients.

Top 9 VM-NGWP best basis vectors



Observations from Examples 1 & 2

- Basis vectors that behave like *oriented edge detectors* are *automatically generated* in the NGWP dictionary.
- NGWP outperformed the other dictionaries on piecewise smooth graph signals
- NGWP performed reasonably well for graph signals sampled on an image containing oriented anisotropic texture patterns (e.g., Barbara's pants)
- NGWP was beaten by the eGHWT on the non-smooth and localized graph signal (e.g., pedestrian volume data on the Toronto street map)
- Potential reason I: NGWP is a direct generalization of the *Shannon* wavelet packet dictionaries, i.e., their "frequency" domain support is localized and well controlled while the "time" domain support is not compact.
- Potential reason II: The eGHWT tends to have better performance with oscillatory non-smooth signals in general compared to the other transforms [Shao & Saito, 2019].

Observations from Examples 1 & 2

- Basis vectors that behave like *oriented edge detectors* are *automatically generated* in the NGWP dictionary.
- NGWP outperformed the other dictionaries on piecewise smooth graph signals
- NGWP performed reasonably well for graph signals sampled on an image containing oriented anisotropic texture patterns (e.g., Barbara's pants)
- NGWP was beaten by the eGHWT on the non-smooth and localized graph signal (e.g., pedestrian volume data on the Toronto street map)
- Potential reason I: NGWP is a direct generalization of the *Shannon* wavelet packet dictionaries, i.e., their "frequency" domain support is localized and well controlled while the "time" domain support is not compact.
- Potential reason II: The eGHWT tends to have better performance with oscillatory non-smooth signals in general compared to the other transforms [Shao & Saito, 2019].

Observations from Examples 1 & 2

- Basis vectors that behave like *oriented edge detectors* are *automatically generated* in the NGWP dictionary.
- NGWP outperformed the other dictionaries on piecewise smooth graph signals
- NGWP performed reasonably well for graph signals sampled on an image containing oriented anisotropic texture patterns (e.g., Barbara's pants)
- NGWP was beaten by the eGHWT on the non-smooth and localized graph signal (e.g., pedestrian volume data on the Toronto street map)
- Potential reason I: NGWP is a direct generalization of the *Shannon* wavelet packet dictionaries, i.e., their "frequency" domain support is localized and well controlled while the "time" domain support is not compact.
- Potential reason II: The eGHWT tends to have better performance with oscillatory non-smooth signals in general compared to the other transforms [Shao & Saito, 2019].

Observations from Examples 1 & 2

- Basis vectors that behave like *oriented edge detectors* are *automatically generated* in the NGWP dictionary.
- NGWP outperformed the other dictionaries on piecewise smooth graph signals
- NGWP performed reasonably well for graph signals sampled on an image containing oriented anisotropic texture patterns (e.g., Barbara's pants)
- NGWP was beaten by the eGHWT on the non-smooth and localized graph signal (e.g., pedestrian volume data on the Toronto street map)
- Potential reason I: NGWP is a direct generalization of the *Shannon* wavelet packet dictionaries, i.e., their "frequency" domain support is localized and well controlled while the "time" domain support is not compact.
- Potential reason II: The eGHWT tends to have better performance with oscillatory non-smooth signals in general compared to the other transforms [Shao & Saito, 2019].

Observations from Examples 1 & 2

- Basis vectors that behave like *oriented edge detectors* are *automatically generated* in the NGWP dictionary.
- NGWP outperformed the other dictionaries on piecewise smooth graph signals
- NGWP performed reasonably well for graph signals sampled on an image containing oriented anisotropic texture patterns (e.g., Barbara's pants)
- NGWP was beaten by the eGHWT on the non-smooth and localized graph signal (e.g., pedestrian volume data on the Toronto street map)
- Potential reason I: NGWP is a direct generalization of the *Shannon* wavelet packet dictionaries, i.e., their "frequency" domain support is localized and well controlled while the "time" domain support is not compact.
- Potential reason II: The eGHWT tends to have better performance with oscillatory non-smooth signals in general compared to the other transforms [Shao & Saito, 2019].

Observations from Examples 1 & 2

- Basis vectors that behave like *oriented edge detectors* are *automatically generated* in the NGWP dictionary.
- NGWP outperformed the other dictionaries on piecewise smooth graph signals
- NGWP performed reasonably well for graph signals sampled on an image containing oriented anisotropic texture patterns (e.g., Barbara's pants)
- NGWP was beaten by the eGHWT on the non-smooth and localized graph signal (e.g., pedestrian volume data on the Toronto street map)
- Potential reason I: NGWP is a direct generalization of the *Shannon* wavelet packet dictionaries, i.e., their "frequency" domain support is localized and well controlled while the "time" domain support is not compact.
- Potential reason II: The eGHWT tends to have better performance with oscillatory non-smooth signals in general compared to the other transforms [Shao & Saito, 2019].

Outline

Summary and Future Projects

- Developed a *natural* method to generate the *smooth graph wavelet packet (NGWP) dictionary*.
- Used the *hierarchical bipartitioning* of the dual $G^\star$ consisting of the GL eigenvectors as its nodes
- Used *varimax rotation* to get wavelet packet vectors well-localized on G.
- How to reduce computational complexity of $O(n^3)$
- How to proceed more precise approximation theoretic statements
- How to deal with the GL eigenvectors corresponds to *multiple eigenvalues*
- Can we incorporate clustering with *overlaps* on either G ($\Longrightarrow$ *graph local cosines*) or $G^\star$ ($\Longrightarrow$ *graph Meyer wavelet packets*) ?

Summary and Future Projects

- Developed a *natural* method to generate the *smooth graph wavelet packet (NGWP) dictionary*.
- Used the *hierarchical bipartitioning* of the dual $G^\star$ consisting of the GL eigenvectors as its nodes
- Used *varimax rotation* to get wavelet packet vectors well-localized on G.
- How to reduce computational complexity of $O(n^3)$
- How to proceed more precise approximation theoretic statements
- How to deal with the GL eigenvectors corresponds to *multiple eigenvalues*
- Can we incorporate clustering with *overlaps* on either G ($\Longrightarrow$ *graph local cosines*) or $G^\star$ ($\Longrightarrow$ *graph Meyer wavelet packets*) ?

References

- J. Irion & N. Saito: "Hierarchical graph Laplacian eigen transforms," *JSIAM Letters*, vol.6, pp.21–24, 2014.
- J. Irion & N. Saito: "The generalized Haar-Walsh transform," in *Proc. 2014 IEEE Workshop on Statistical Signal Processing*, pp. 472–475, 2014.
- J. Irion & N. Saito: "Applied and computational harmonic analysis on graphs and networks," in *Wavelets and Sparsity XVI, Proc. SPIE 9597*, Paper # 95971F, 2015.
- J. Irion & N. Saito: "Efficient approximation and denoising of graph signals using the multiscale basis dictionaries," *IEEE Trans. Signal and Inform. Process. Netw.*, vol. 3, no. 3, pp. 607–616, 2017.
- N. Saito: "How can we naturally order and organize graph Laplacian eigenvectors?" in *Proc. 2018 IEEE Workshop on Statistical Signal Processing*, pp. 483–487, 2018.
- A. Cloninger & S. Steinerberger: "On the dual geometry of Laplacian eigenfunctions," *Experimental Mathematics*, `DOI:10.1080/10586458.2018.1538911`, 2018.
- H. Li & N. Saito: "Metrics of graph Laplacian eigenvectors," in *Wavelets and Sparsity XVIII, Proc. SPIE 11138*, Paper #11138K, 2019.
- A. Cloninger, H. Li, & N. Saito: "Natural graph wavelet packet dictionaries," `arXiv:2009.09020`, 2020.

Thank you very much for your attention!