# The extended Generalized Haar-Walsh Transform and applications

Yiqun Shao and Naoki Saito

Department of Mathematics, University of California, One Shields Avenue, Davis, CA 95616, USA

## ABSTRACT

Extending computational harmonic analysis tools from the classical setting of regular lattices to the more general setting of graphs and networks is very important and much research has been done recently. Our previous Generalized Haar-Walsh Transform (GHWT) is a multiscale transform for signals on graphs, which is a generalization of the classical Haar and Walsh-Hadamard Transforms. This article proposes the extended Generalized Haar-Walsh Transform (eGHWT). The eGHWT and its associated best-basis selection algorithm for graph signals will significantly improve the performance of the previous GHWT with the similar computational cost, $O(N \log N)$ where $N$ is the number of nodes of an input graph. While the previous GHWT/best-basis algorithm seeks the most suitable orthonormal basis for a given task among more than $(1.5)^N$ possible bases, the eGHWT/best-basis algorithm can find a better one by searching through more than $0.618 \cdot (1.84)^N$ possible bases. This article describes the details of the eGHWT/basis-basis algorithm and demonstrates its superiority using several examples including genuine graph signals as well as conventional digital images viewed as graph signals.

**Keywords:** Multiscale basis dictionaries, wavelets on graphs, graph signal processing, adapted time-frequency analysis, the best-basis algorithm

## 1. INTRODUCTION

In recent years, research on graphs and networks is experiencing rapid growth due to a confluence of several trends in science and technology: the advent of new sensors, measurement technologies, and social network infrastructure has provided huge opportunities to visualize complicated interconnected network structures, record data of interest at various locations in such networks, analyze such data, and make inferences and diagnostics. We can easily observe such network-based problems in truly diverse fields: biology and medicine (e.g., connectomes); computer science (e.g., social networks); electrical engineering (e.g., sensor networks); hydrology and geology (e.g., ramified river networks); and civil engineering (e.g., road networks), to name just a few. Consequently, there is an explosion of interest and demand to analyze data sampled on such graphs and networks, which are often called "network data analysis" or "graph signal processing"; see e.g., recent books[1–4] and survey articles,[5,6] to see the evidence of this trend. This trend has forced the signal processing community to extend classical techniques on regular domains to the setting of graphs. Much efforts have been done to develop wavelet transforms for signals on graphs (or the so-called graph signals).[7–15] Comprehensive reviews of transforms for signals on graphs have also been written.[5,6]

The Generalized Haar-Walsh Transform (GHWT),[16–18] developed by Irion and Saito, has achieved superior results over other transforms in terms of both approximation and denoising of signals on graphs (or graph signals for short). Using a recursive partitioning of the graph and successive averaging and differencing operations, the GHWT generates two overcomplete dictionaries of orthonormal bases (ONBs). They are reordered versions of each other and are called *coarse-to-fine* (c2f) and *fine-to-coarse* (f2c) GHWT dictionaries. The *best-basis algorithm*, which is extended for the graph setting from what Coifman and Wickerhauser developed for the regular lattices,[19] is then applied to those two dictionaries to find the most suitable ONB for a given task (e.g., compression, denoising, ...).

Further author information: (Send correspondence to Y.S.)
Y.S.: E-mail: yqshao@math.ucdavis.edu, URL: `https://www.math.ucdavis.edu/~yqshao`
N.S.: E-mail: saito@math.ucdavis.edu, URL: `https://www.math.ucdavis.edu/~saito`

In the mean time, Thiele and Villemoes[20] proposed an algorithm to find the best basis among discretized and rescaled Walsh functions for an input 1D (non-graph) signal of dyadic length. We modify and extend this algorithm to GHWT to develop the *extended GHWT* (eGHWT). The eGHWT can search in a much larger set of ONBs that include those searched by the GHWT with similar computational cost.

In Section 2, we introduce some basic concepts of graph signal processing. Then the GHWT/best-basis algorithm is reviewed. In Section 3, we provide an overview of the eGHWT. Example on 6-nodes signal explaining the eGHWT and illustrating the difference from the GHWT is given. Section 4 demonstrates the superiority of the eGHWT over the GHWT (including the graph Haar and Walsh bases) using real datasets.

## 2. THE GENERALIZED HAAR-WALSH TRANSFORM (GHWT)

In this section, we will first establish our notation and define some important quantities which will be used throughout this article. Then, we will review the Generalized Haar-Walsh Transform (GHWT).[16–18] It is a multiscale transform for graph signals and a *true* generalization of the classical Haar and Walsh-Hadamard transforms: if an input graph is a simple path graph whose number of nodes is dyadic, then the GHWT reduces to the classical counterpart *exactly*.

### 2.1 Notations and Definitions

A graph is a pair $G = (V, E)$, where $V = V(G) = \{v_1, v_2, \ldots, v_N\}$ is the vertex (or node) set of $G$, and $E = E(G) = \{e_1, e_2, \ldots, e_M\}$ is the edge set, where each edge connects two nodes $v_i, v_j$ for some $1 \leq i \neq j \leq N$. We only deal with finite $N$ and $M$ in this article. For simplicity, we often write $i$ instead of $v_i$.

An edge connecting a node $i$ and itself is called a *loop*. If there exists more than one edge connecting some $i, j$, then they are called *multiple edges*. A graph having loops or multiple edges is called a *multiple graph* (or multigraph); a graph with neither of these is called a *simple graph*. A *directed graph* is a graph in which edges have orientations while *undirected graph* is a graph in which edges do not have orientations. If each edge $e \in E$ has a weight (normally nonnegative), then $G$ is called a *weighted graph*. A *path* from $i$ to $j$ in a graph $G$ is a subgraph of $G$ consisting of a sequence of distinct nodes starting with $i$ and ending with $j$ such that consecutive nodes are adjacent. A path starting from $i$ that returns to $i$ (but is not a loop) is called a *cycle*. For any two distinct nodes in $V$, if there is a path connecting them, then such a graph is said to be *connected*. In this article, we only consider *undirected weighted simple connected graph*.

Sometimes, each node is associated with spatial coordinates in $\mathbb{R}^d$. For example, if we want to analyze a network of sensors and build a graph whose nodes represent the sensors under consideration, then these nodes have the spatial coordinates in $\mathbb{R}^2$ or $\mathbb{R}^3$ indicating their current locations. In that case, we write $\boldsymbol{x}[i] \in \mathbb{R}^d$ for the location of node $i$. Let $\boldsymbol{f} = (f[1], \ldots, f[N])^{\mathsf{T}} \in \mathbb{R}^N$ be a data vector (often called a *graph signal*) where $f[i] \in \mathbb{R}$ is the value measured at the node $i$ of the graph.

We now discuss several matrices associated with undirected simple graphs. The information in both $V$ and $E$ is captured by the *edge weight matrix* $W(G) \in \mathbb{R}^{N \times N}$, where $W_{ij} \geq 0$ is the edge weight between nodes $i$ and $j$. In an unweighted graph, this is restricted to be either 0 or 1, depending on whether nodes $i$ and $j$ are adjacent, and we may refer to $W(G)$ as an *adjacency matrix*. In a weighted graph, $W_{ij}$ indicates the *affinity* between $i$ and $j$. In either case, since $G$ is undirected, $W(G)$ is a symmetric matrix. We then define the *degree matrix* $D(G) := \operatorname{diag}(d_1, \ldots, d_N)$, $d_i := \sum_j W_{ij}$. With this in place, we are now able to define the *(unnormalized) Laplacian* matrix, *random-walk normalized Laplacian* matrix, and *symmetric normalized Laplacian* matrix, as $L(G) := D(G) - W(G)$, $L_{\mathrm{rw}}(G) := D(G)^{-1}L(G)$, $L_{\mathrm{sym}}(G) := D(G)^{-1/2}L(G)D(G)^{-1/2}$, respectively. See[21] for the details of the relationship between these three matrices and their spectral properties. We use $0 = \lambda_0 < \lambda_1 \leq \ldots \leq \lambda_{N-1}$ to denote the sorted Laplacian eigenvalues and $\boldsymbol{\phi}_0, \boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{N-1}$ to denote their corresponding eigenvectors, where the specific Laplacian matrix to which they refer will be clear from either context or subscripts. Which version of Laplacian matrices should be used depends on applications. For example, von Luxburg[21] recommends the use of the eigenvectors of $L_{\mathrm{rw}}$ for graph partitioning. To reduce the computational complexity (as we did for the GHWT construction), we only use the *Fiedler vector*,[22] i.e., the eigenvector $\boldsymbol{\phi}_1$ corresponding to the smallest nonzero eigenvalue $\lambda_1$, to bipartition a given subgraph in this article.

Below, we will briefly review the construction of the GHWT, which consists of two main steps: 1) recursively bipartitioning an input graph; and 2) generating a full ONB on each level of the graph partitioning based on simple yet careful averaging and differencing operations. We will review each step below.

### 2.2 Recursive Partitioning of Graphs

The foundation upon which the GHWT (and eGHWT) is constructed is a *binary partition tree* (also known as a *hierarchical bipartition tree*) of an input graph $G(V, E)$: a set of tree-structured subgraphs of $G$ constructed by recursively bipartitioning $G$. This bipartitioning operation ideally splits each subgraph into two smaller subgraphs that are roughly equal in size while keeping tightly-connected nodes grouped together. We typically use the spectral graph partitioning with the Fiedler vectors of the $L_{\mathrm{rw}}$ matrices of subgraphs for this bipartitioning. More specifically, let $G_k^j$ denote the $k$th subgraph on level $j$ of the binary partition tree of $G$. Note $G_0^0 = G$ and level $j = 0$ represents the root node of this tree. The two children of $G_k^j$ in the tree, $G_{k'}^{j+1}$ and $G_{k'+1}^{j+1}$, are obtained through partitioning $G_k^j$ using the Fiedler vector of $L_{\mathrm{rw}}(G_k^j)$. The graph partitioning is recursively performed until each subgraph corresponding to the leaf contains only one node. We note that any graph cut methods other than spectral graph partitioning can be used as well as long as the binary partition tree of $G$ is constructed.

### 2.3 Overcomplete Dictionaries of Bases

After the binary partition tree of the input graph with depth $j_{\max}$ is generated, an overcomplete dictionary of basis vectors is composed by Algorithm 1 shown below. Each basis vector is denoted as $\boldsymbol{\psi}_{k,l}^j$, where $j$ denotes the level, $k$ denotes the region and $l$ denotes the tag. The tag $l$ of specifies the sequence of average and difference operations by which $\boldsymbol{\psi}_k^j$ was generated. Generally speaking, larger $l$ values indicate more oscillation, with exceptions when imbalances occur in the recursive partitioning. We refer to basis vectors with tag $l = 0$ as *scaling vectors*, those with tag $l = 1$ as *Haar vectors*, and those with tag $l \geq 2$ as *Walsh vectors*.

Several observations are in order. First, the basis vectors on each level are localized. In other words, $\boldsymbol{\psi}_{k,l}^j$ is supported on $V(G_k^j)$. If $V(G_k^j) \cap V(G_{k'}^{j'}) = \emptyset$, then the basis vectors $\{\boldsymbol{\psi}_{k,l}^j\}_l$ and $\{\boldsymbol{\psi}_{k',l'}^{j'}\}_{l'}$ are mutually orthogonal. Second, the basis vectors corresponding to $G_k^j$ span the same linear subspace as the union of those corresponding to $G_{k'}^{j+1}$ and $G_{k'+1}^{j+1}$, where $G_{k'}^{j+1}$ and $G_{k'+1}^{j+1}$ are the two subgraphs of $G_k^j$. Third, the depth of the dictionary is the same as the binary partition tree, which is approximately $O(\log N)$ if the tree is nearly balanced. There are $N$ vectors on each level, so the total number of basis vectors is approximately $O(N \log N)$. In addition, we can arrange these basis vectors by region, which we call the *coarse-to-fine* (c2f) dictionary, or by tag, which we call the *fine-to-coarse* (f2c) dictionary.[16–18] The c2f dictionary corresponds to a collection of basis vectors by recursively partitioning the "time" domain information of the input graph signal while the f2c dictionary corresponds to those by recursively partitioning the "frequency" (or "sequency") domain information of the input graph signal. Each dictionary contains more than $(1.5)^N$ choosable ONBs; see, e.g., Thiele and Villemoes[20] for the details on this number. Note, however, that exceptions can occur when the recursive partitioning generates a highly imbalanced tree. Figures 1 and 2 show the examples of these dictionaries for a simple path graph consisting of six nodes.

### 2.4 The Previous Best-Basis Algorithm in the GHWT

To select the basis from a dictionary of wavelet packets that is "best" for approximation/compression, a *best-basis algorithm* is required. The previous best-basis algorithm in GHWT is a straightforward generalization of Coifman and Wickerhauser,[19] which was developed for non-graph signals of dyadic length. The algorithm requires a real-valued cost function $\mathcal{J}$. In the GHWT case, the algorithm works as long as $\mathcal{J}$ is nonnegative and additive [*] of the form $\mathcal{J}(\boldsymbol{c}) := \sum_i g(c[i])$ where $\boldsymbol{c}$ is the expansion coefficients of an input graph signal on each region. For example, if one wants to promote sparsity in graph signal representation or approximation, one can set this $g$ function as $g(x) = |x|^p$, $0 < p \leq 1$, The algorithm aims to find the best basis whose coefficients minimize $\mathcal{J}$. Algorithm 2 describes the details of this procedure. This algorithm initiates the best basis as the

[*]The additivity property can be dropped in principle by following the work of Saito and Coifman on the local regression basis[23]

**Algorithm 1:** Generating GHWT Dictionary[16–18]

```
Input: A binary partition tree $\{G_k^j\}$ of the graph $G$, $0 \le j \le j_{\max}$ and $0 \le k < K^j$. $N_k^j := |V(G_k^j)|$.
       $K^j$ denotes the number of subgraphs on level $j$.
Output: An overcomplete dictionary of basis vectors $\{\boldsymbol{\psi}_{k,l}^j\}$
for $k = 0, \dots, N-1$ do                    // Basis vectors on level $j_{\max}$ are unit vectors
    $\boldsymbol{\psi}_{k,0}^{j_{\max}} \leftarrow 1_{V(G_k^{j_{\max}})} \in \mathbb{R}^N$
end
for $j = j_{\max}, \dots, 1$ do               // Compose basis vectors on level $j-1$ from level $j$
    for $k = 0, \dots, K^{j-1} - 1$ do
        $\boldsymbol{\psi}_{k,0}^{j-1} \leftarrow 1_{V(G_k^{j-1})} / \sqrt{N_k^{j-1}}$        // Compute the scaling vector
        // Basis vectors supported on $V(G_k^{j-1})$ are computed from those on $V(G_{k'}^j)$ and
        $V(G_{k'+1}^j)$.  $G_{k'}^j$ and $G_{k'+1}^j$ are the two subgraphs of $G_k^{j-1}$
        if $N_k^{j-1} > 1$ then
            $\boldsymbol{\psi}_{k,1}^{j-1} \leftarrow \frac{N_{k'+1}^j \sqrt{N_{k'}^j} \boldsymbol{\psi}_{k',0}^j - N_{k'}^j \sqrt{N_{k'+1}^j} \boldsymbol{\psi}_{k'+1,0}^j}{\sqrt{N_{k'}^j (N_{k'+1}^j)^2 + N_{k'+1}^j (N_{k'}^j)^2}}$        // Compute the Haar vector
        end
        if $N_k^{j-1} > 2$ then
            for $l = 1, \dots, 2^{j_{\max}-j} - 1$ do        // Compute the Walsh-like vectors
                if both subregions $k'$ and $k'+1$ have a basis vector with tag $l$ then
                    $\boldsymbol{\psi}_{k,2l}^{j-1} \leftarrow (\boldsymbol{\psi}_{k',l}^j + \boldsymbol{\psi}_{k'+1,l}^j)/\sqrt{2}$;
                    $\boldsymbol{\psi}_{k,2l+1}^{j-1} \leftarrow (\boldsymbol{\psi}_{k',l}^j - \boldsymbol{\psi}_{k'+1,l}^j)/\sqrt{2}$;
                else if (without loss of generality) only subregion $k'$ has a basis vector with tag $l$ then
                    $\boldsymbol{\psi}_{k,2l}^{j-1} \leftarrow \boldsymbol{\psi}_{k',l}^j$
                else if Neither subregion has a basis vector with tag $l$ then
                    Do nothing
            end
        end
    end
end
```

**Algorithm 2:** The best-basis algorithm of Coifman-Wickerhauser[19] tailored for the GHWT[16–18]

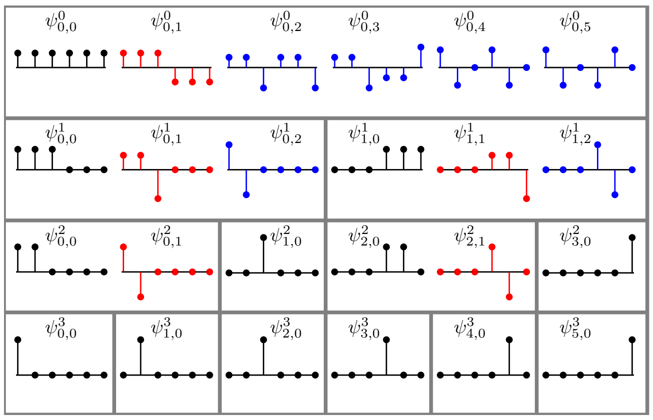
```
Input: An overcomplete dictionary of basis vectors $\{\boldsymbol{\psi}^j_{k,l}\}$ from Algorithm 1, $0 \le j \le j_{\max}$ and
       $0 \le k < K^j$ where $K^j$ denotes the number of subgraphs on level $j$; an additive cost function
       $\mathcal{J}(\boldsymbol{c}) = \sum_{i=1}^N g(c[i])$; an input graph signal $\boldsymbol{f} \in \mathbb{R}^N$.
Output: The best basis $B(G)$; the minimal cost $A(G)$.
for $k = 0, \dots, N-1$ do                                // Initialize the best basis on level $j_{\max}$
    $B(G^{j_{\max}}_k) \leftarrow \{\boldsymbol{\psi}^{j_{\max}}_{k,0}\}$;
    $A(G^{j_{\max}}_k) = g(\langle \boldsymbol{\psi}^{j_{\max}}_{k,0}, \boldsymbol{f} \rangle)$
end
for $j = j_{\max} - 1, \dots, 0$ do                            // Update the best basis upwards
    for $k = 0, \dots, K^j$ do
        $B(G^j_k) \leftarrow \{\boldsymbol{\psi}^j_{k,l}\}$;
        $A(G^j_k) \leftarrow \sum_l g(\langle \boldsymbol{\psi}^j_{k,l}, \boldsymbol{f} \rangle)$;
        if $G^j_k$ is split into two subgraphs $G^{j+1}_{k'}$ and $G^{j+1}_{k'+1}$ then
            if $A(G^j_k) \ge A(G^{j+1}_{k'}) + A(G^{j+1}_{k'+1})$ then
                $B(G^j_k) \leftarrow B(G^{j+1}_{k'}) \cup B(G^{j+1}_{k'+1})$;
                $A(G^j_k) \leftarrow A(G^{j+1}_{k'}) + A(G^{j+1}_{k'+1})$
            end
        end
    end
end
$B(G) \leftarrow B(G^0_0)$;
$A(G) \leftarrow A(G^0_0)$;
```

whole set of vectors at bottom level of the dictionary. Then it proceeds upwards, comparing the cost of the expansion coefficients corresponding to two children subgraphs to the cost of those of their parent subgraph. The best basis is updated if the cost of the parent subgraph is smaller than that of its children subgraphs. The algorithm continues until it reaches the top (i.e., the root) of the binary partition tree (i.e., the dictionary). Th c2f and f2c dictionaries are searched separately to obtain two sets of the best bases, among which the one with smaller cost is chosen as the final best basis of the GHWT dictionaries. We note here that the graph Haar basis is selectable *only in the f2c dictionary* while the graph Walsh basis is selectable in either dictionary.

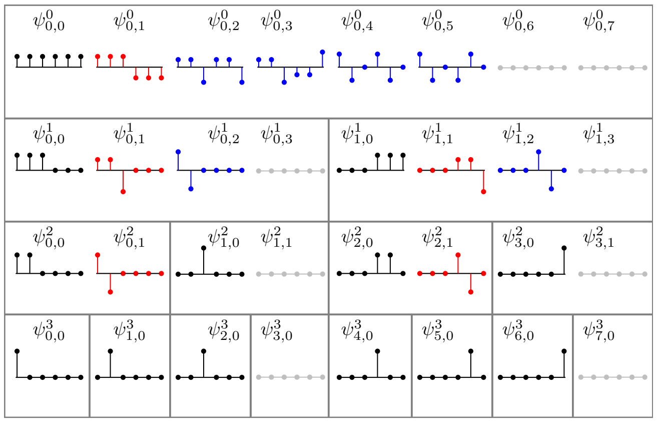
## 3. THE EXTENDED GENERALIZED HAAR-WALSH TRANSFORM (eGHWT)

In this section, we describe the *extended* GHWT, i.e., our new best-basis algorithm on the GHWT dictionaries, which *simultaneously* considers the "time" domain split *and* "frequency" (or "sequency") domain split of an input graph signal. This transform will allow us to deploy the modified best-basis algorithm that can select the best ONB for one's task (e.g., efficient approximation, denoising, etc.) among a much larger set ($> 0.618 \cdot (1.84)^N$) of ONBs that the c2f and f2c GHWT dictionaries would provide ($> (1.5)^N$). The previous best-basis algorithm only searches through the c2f dictionary and f2c dictionary separately, but this new method combined them together to search through them simultaneously.

Thiele and Villemoes[20] proposed an algorithm to find the best basis among the ONBs of $\mathbb{R}^N$ consisting of discretized rescaled Walsh functions for an input 1D (non-graph) signal of length $N$, where $N$ must be a dyadic integer. Their algorithm operates in the time-frequency plane by constructing a tiling of minimal cost among all possible tilings with dyadic rectangles of area one. Here we adapt their method to our graph setting that does not have to be dyadic. In addition, its generalization for 2D signals developed by Lindberg and Villemoes[24] can be generalized to the 2D eGHWT by considering the tensor product of binary partition trees as we will discuss more in Section 4.2.


$\psi^0_{0,0}$ $\psi^0_{0,1}$ $\psi^0_{0,2}$ $\psi^0_{0,3}$ $\psi^0_{0,4}$ $\psi^0_{0,5}$
$\psi^1_{0,0}$ $\psi^1_{0,1}$ $\psi^1_{0,2}$ $\psi^1_{1,0}$ $\psi^1_{1,1}$ $\psi^1_{1,2}$
$\psi^2_{0,0}$ $\psi^2_{0,1}$ $\psi^2_{1,0}$ $\psi^2_{2,0}$ $\psi^2_{2,1}$ $\psi^2_{3,0}$
$\psi^3_{0,0}$ $\psi^3_{1,0}$ $\psi^3_{2,0}$ $\psi^3_{3,0}$ $\psi^3_{4,0}$ $\psi^3_{5,0}$

(a) The c2f GHWT dictionary


$\psi^0_{0,0}$ $\psi^0_{0,1}$ $\psi^0_{0,2}$ $\psi^0_{0,3}$ $\psi^0_{0,4}$ $\psi^0_{0,5}$ $\psi^0_{0,6}$ $\psi^0_{0,7}$
$\psi^1_{0,0}$ $\psi^1_{0,1}$ $\psi^1_{0,2}$ $\psi^1_{0,3}$ $\psi^1_{1,0}$ $\psi^1_{1,1}$ $\psi^1_{1,2}$ $\psi^1_{1,3}$
$\psi^2_{0,0}$ $\psi^2_{0,1}$ $\psi^2_{1,0}$ $\psi^2_{1,1}$ $\psi^2_{2,0}$ $\psi^2_{2,1}$ $\psi^2_{3,0}$ $\psi^2_{3,1}$
$\psi^3_{0,0}$ $\psi^3_{1,0}$ $\psi^3_{2,0}$ $\psi^3_{3,0}$ $\psi^3_{4,0}$ $\psi^3_{5,0}$ $\psi^3_{6,0}$ $\psi^3_{7,0}$

(b) The relabeled c2f GHWT dictionary

Figure 1: (a) The c2f GHWT dictionary on the simple path $P_6$. Stem plots with black, red, blue colors correspond to the scaling, Haar, and Walsh vectors, respectively. (b) The relabeled c2f GHWT dictionary by Algorithm 3 applied to the c2f GHWT dictionary shown in (a). The gray stem plots indicate the "fictitious" (or "non-existent") nodes newly generated by Algorithm 3.

### 3.1 Relabeling Region Indices

Before the new best-basis algorithm is applied, the binary partition tree of an input graph and the overcomplete dictionary of basis vectors are formed in the same way as the c2f GHWT dictionary. Then to deploy the new best-basis algorithm, the region index $k$ of $G^j_k$ and $\boldsymbol{\psi}^j_{k,l}$ needs to be *relabeled*. Previously, on level $j$, the region index $k$ takes all the integer values in $[0, K^j)$ where $K^j$ is the total number of subgraphs (or regions) on level $j$. After relabeling, $k$ takes an integer values in $[0, 2^j - 1]$ according to its location in the binary tree. The whole procedure is described by Algorithm 3. Then the region indices of the basis vectors $\{\boldsymbol{\psi}^j_{k,l}\}$ supported on the subgraph $G^j_k$ are also relabeled accordingly. Figure 1b shows the result of Algorithm 3 applied to the c2f GHWT

**Algorithm 3:** Relabeling the GHWT Dictionary

```
Input: A binary partition tree denoted by $\{G^j_k\}$, $0 \le k < K^j$, $0 \le j < j_{\max}$, $K^j$ denotes the number of
       subgraphs on level $j$
Output: The same binary partition tree $\{G^j_k\}$ with region index $k$ relabeled
// On level 0, there is only one region $G^0_0$, so no relabeling is required.
for $j = 1, \ldots, j_{\max}$ do
    if $G^{j-1}_k$ is split into $G^j_{k'}$ and $G^j_{k'+1}$ then
        The two subgraphs are relabeled as $G^j_{2k}$ and $G^j_{2k+1}$
    else if $G^{j-1}_k$ is kept as $G^j_{k'}$ then                // the subgraph contains only one node.
        The subgraph is relabeled as $G^j_{2k}$
end
```

dictionary shown in Figure 1a on a simple path graph with $N = 6$.

### 3.2 The New Best-Basis (eGHWT) Algorithm

We can then apply Algorithm 4 to search for the best basis in the new GHWT dictionary. This whole procedure is called the *eGHWT*.

[*]An *associative array* is an abstract data type composed of a collection of (key, value) pairs such that each possible key appears at most once in the collection.

[†]$((j,k,l), g(\langle \boldsymbol{\psi}^j_{k,l}, \boldsymbol{f} \rangle))$ is a pair of (key, value) of the associative array $A_0$. Here we use $(j,k,l) \in A_0$ to denote that $(j,k,l)$ is a valid key of $A_0$. Therefore, $(j,k,l) \in A_0$ if and only if $\boldsymbol{\psi}^j_{k,l}$ exists. Since we relabeled $\boldsymbol{\psi}^j_{k,l}$, there is no

**Algorithm 4:** The New Best-Basis (eGHWT) Algorithm

```
Input: The dictionary of basis vectors $\{\boldsymbol{\psi}_{k,l}^j\}$ from Algorithm 3; an additive cost function
        $\mathcal{J}(\boldsymbol{c}) = \sum_{i=1}^{N} g(c[i])$; an input graph signal $\boldsymbol{f} \in \mathbb{R}^N$.
Output: The best basis $B$.
Initialize the associative array* $A_0$ as the expansion coefficients† of the dictionary evaluated through
  $g(\cdot)$, i.e., $A_0[j,k,l] = g(\langle \boldsymbol{\psi}_{k,l}^j, \boldsymbol{f} \rangle)$;
Initialize another associative array $I_0$ by $I_0[j,k,l] = 1$ if $(j,k,l) \in A_0$;
for $m = 0, \dots, j_{\max} - 1$ do                                   // Compute $A_{m+1}$ and $I_{m+1}$
    Initialize $A_{m+1}$ and $I_{m+1}$ as empty associative arrays;
    for $(j,k,l) \in A_m$ with $j < j_{\max}$ and $l \equiv 0 \pmod 2$ do
        if $(A_m[j,k,l] + A_m[j,k,l+1]) \le (A_m[j+1,2k,l/2] + A_m[j+1,2k+1,l/2])$ then
            $I_{m+1}[j,k,l/2] \leftarrow 0$;
            $A_{m+1}[j,k,l/2] \leftarrow (A_m[j,k,l] + A_m[j,k,l+1])$;
        else
            $I_{m+1}[j,k,l/2] \leftarrow 1$;
            $A_{m+1}[j,k,l/2] \leftarrow (A_m[j+1,2k,l/2] + A_m[j+1,2k+1,l/2])$;
        end
        // Any of $A_m[j,k,l+1], A_m[j+1,2k,l/2], A_m[j+1,2k+1,l/2]$ will be replaced by 0 in
           the above steps if it does not exist.
    end
end
$I \leftarrow I_{j_{\max}}$;
for $m = j_{\max} - 1, \dots, 0$ do                          // Recover the best basis from $\{I_m\}$
    Initialize $I_{\text{temp}}$ as an empty associative array;
    for $(j,k,l) \in I$ do
        if $I[j,k,l] = 0$ then
            $I_{\text{temp}}[j,k,l] \leftarrow I_m[j,k,2l]$ if $(j,k,2l) \in I_m$;
            $I_{\text{temp}}[j,k,l] \leftarrow I_m[j,k,2l+1]$ if $(j,k,2l+1) \in I_m$;
        else
            $I_{\text{temp}}[j+1,2k,l] \leftarrow I_m[j+1,2k,l]$ if $(j+1,2k,l) \in I_m$;
            $I_{\text{temp}}[j+1,2k+1,l] \leftarrow I_m[j+1,2k+1,l]$ if $(j+1,2k+1,l) \in I_m$;
        end
    end
    $I \leftarrow I_{\text{temp}}$;
end
$B \leftarrow \{\}$;                                              // Initialize $B$ as an empty set
Add $\boldsymbol{\psi}_{k,l}^j$ into $B$ if $(j,k,l) \in I$.                       // $B$ is the best basis
```

Several remarks on the algorithm are in order:

- $A_m[j,k,l]$ is the minimal cost of ONBs in a linear subspace. Generally, $A_m[j,k,l]$ is computed from the minimal of $A_{m-1}[j,k,2l] + A_{m-1}[j,k,2l+1]$ and $A_{m-1}[j+1,2k,l] + A_{m-1}[j+1,2k+1,l]$. The linear subspace of $A_m[j,k,l]$ is the direct sum of the two linear subspaces corresponding to $A_{m-1}[j,k,2l]$ and $A_{m-1}[j,k,2l+1]$, which is the same as the direct sum of those corresponding to $A_{m-1}[j+1,2k,l]$ and $A_{m-1}[j+1,2k+1,l]$. In other words, when we compute $A_m$ from $A_{m-1}$, we are concatenating linear subspaces.
- The linear subspace of one entry in $A_0$ is one dimensional since it is spanned by one basis vector. In other words, $A_0[j,k,l]$ corresponds to the linear subspace spanned by $\boldsymbol{\psi}^j_{k,l}$.
- $A_{j_{\max}}$ has only one entry $A_{j_{\max}}[0,0,0]$, which corresponds to the whole $\mathbb{R}^n$. Its value is the minimal cost of all the ONBs, i.e., the cost of the best basis.
- If the signal is of dyadic length, then $(A_{m-1}[j,k,2l], A_{m-1}[j,k,2l+1])$ corresponds to splitting the subspace of $A_m[j,k,l]$ in the "frequency" domain in the "time-frequency plane" while $(A_{m-1}[j+1,2k,l], A_{m-1}[j+1,2k+1,l])$ does the split in the "time" domain.
- If the signal is of dyadic length, the eGHWT can select among a much larger set ($> 0.618 \cdot (1.84)^N$) of ONBs than what each of the c2f and f2c GHWT dictionaries would provide ($> (1.5)^N$).[20] The numbers are similar even for non-dyadic cases as long as the partition trees are essentially balanced. The essence of this algorithm is that at each step of the recursive evaluation of the costs of subspaces, it compares the cost of the parent subspace with not only its two children subspaces partitioned in the "frequency" domain (the f2c GHWT does this), but also its two children subspaces partitioned in the "time" domain (the c2f GHWT does this).

## 3.3 The eGHWT illustrated by a graph signal on $P_6$

Let $\boldsymbol{f} = [2,-2,1,3,-1,-2]^{\mathsf{T}} \in \mathbb{R}^6$ be an example graph signal on the simple 6-node path $P_6$ to analyze. The $L^1$-norm is chosen as the cost function. Figure 2 shows that the c2f-GHWT best basis is $\frac{\sqrt{6}}{6}\boldsymbol{\psi}^0_{0,0} + \frac{\sqrt{6}}{6}\boldsymbol{\psi}^0_{0,1} + \frac{2\sqrt{3}}{3}\boldsymbol{\psi}^0_{0,2} + \frac{4\sqrt{3}}{3}\boldsymbol{\psi}^0_{0,3} + 4\boldsymbol{\psi}^0_{0,4} + 0\boldsymbol{\psi}^0_{0,5}$ with cost 8.28 and the f2c-GHWT best basis is $\frac{\sqrt{3}}{3}\boldsymbol{\psi}^1_{0,0} + 0\boldsymbol{\psi}^1_{1,0} + \frac{\sqrt{6}}{3}\boldsymbol{\psi}^1_{0,1} + \sqrt{6}\boldsymbol{\psi}^1_{1,1} + 4\boldsymbol{\psi}^0_{0,4} + 0\boldsymbol{\psi}^0_{0,5}$ with cost 7.84 while Figure 3 demonstrates that the best basis chosen by the eGHWT is $0\boldsymbol{\psi}^2_{0,0} + 1\boldsymbol{\psi}^2_{1,0} + 0\boldsymbol{\psi}^1_{1,0} + \sqrt{6}\boldsymbol{\psi}^1_{1,1} + 4\boldsymbol{\psi}^0_{0,4} + 0\boldsymbol{\psi}^0_{0,5}$ with cost 7.45, which is the smallest among these three best-basis representations. The indices used here are before relabeling for the illustration purpose.

## 4. APPLICATIONS

In this section, several examples using eGHWT on real datasets are given.

## 4.1 Efficient Approximation of Graph Signals

Here we analyze the vehicular traffic volume data on the Toronto road network [†], which contains the most recent 8 peak-hour vehicle volume counts collected at intersections where there are traffic signals. The data was typically collected between the hours of 7:30 am and 6:00 pm, over the period of 03/22/2004–02/28/2018. We generated the road network of Toronto using the street names and intersection coordinates included in the dataset. The graph has $N = 2275$ nodes and $M = 3381$ edges. Figure 4a displays this vehicular volume data where the size of the marker is proportional to the vehicular volume.

In addition to the best basis from eGHWT, the graph Haar basis, the graph Walsh basis, best basis from c2f dictionary and best basis from f2c dictionary are used for comparison purpose. The performance is displayed in Figure 4b. The y-axis denotes the relative approximation error $\|\boldsymbol{f} - \mathcal{P}_n\boldsymbol{f}\|_2 / \|\boldsymbol{f}\|_2$, where $\mathcal{P}_n\boldsymbol{f}$ denotes the

corresponding $\boldsymbol{\psi}^j_{k,l}$ for some triple $(j,k,l)$. In that case, $(j,k,l) \notin A_0$.

[†]https://www.toronto.ca/city-government/data-research-maps/open-data/open-data-catalogue/transportation/#7c8e7c62-7630-8b0f-43ed-a2dfe24aadc9

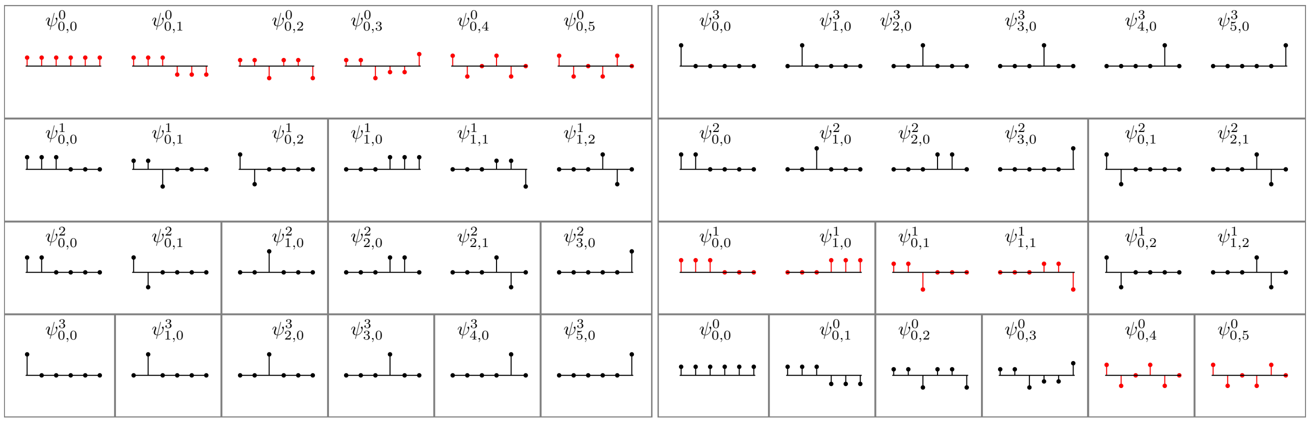


Figure 2: The c2f (a) and f2c (b) GHWT dictionaries for $P_6$. The basis vectors are grouped by region in (a) and by tag in (b). The best basis vectors obtained by GHWT in each dictionary for the input signal $\boldsymbol{f} = [2, -2, 1, 3, -1, -2]^{\mathsf{T}}$ are indicated by red.

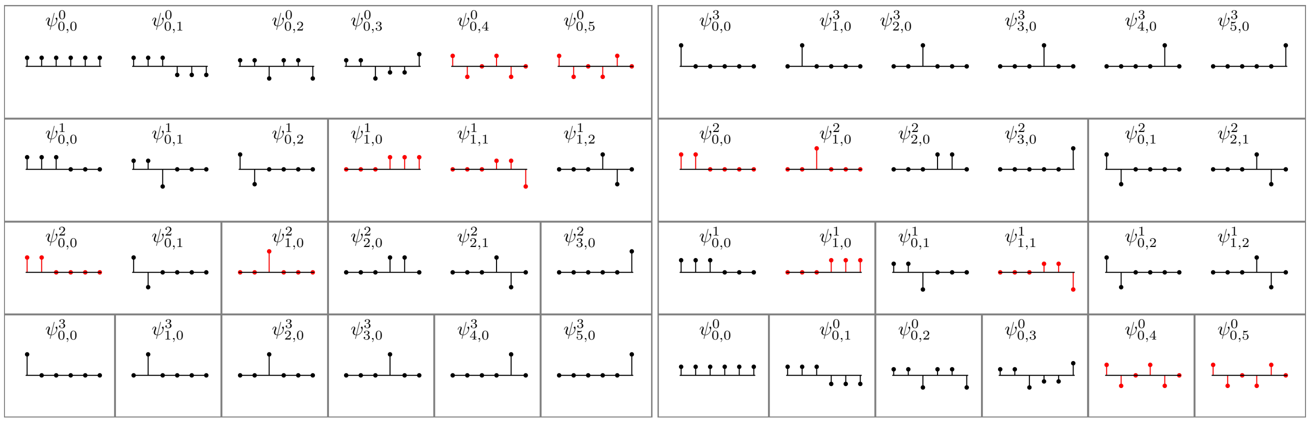
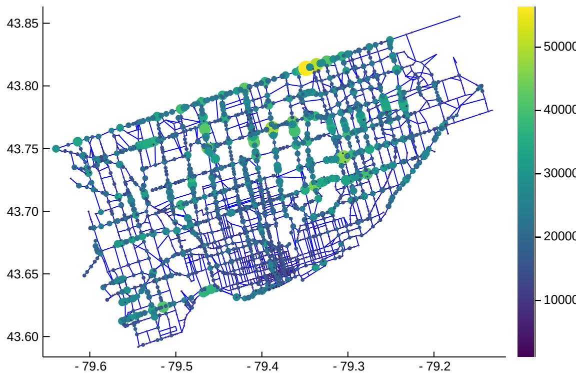
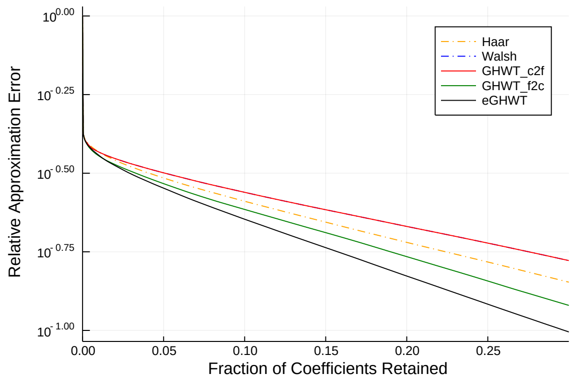


Figure 3: The best basis vectors for the signal $\boldsymbol{f} = [2, -2, 1, 3, -1, -2]^{\mathsf{T}}$, selected by the eGHWT are indicated by red in the c2f GHWT dictionary (a) and f2c GHWT dictionary (b). Note the orthogonality of these vectors.

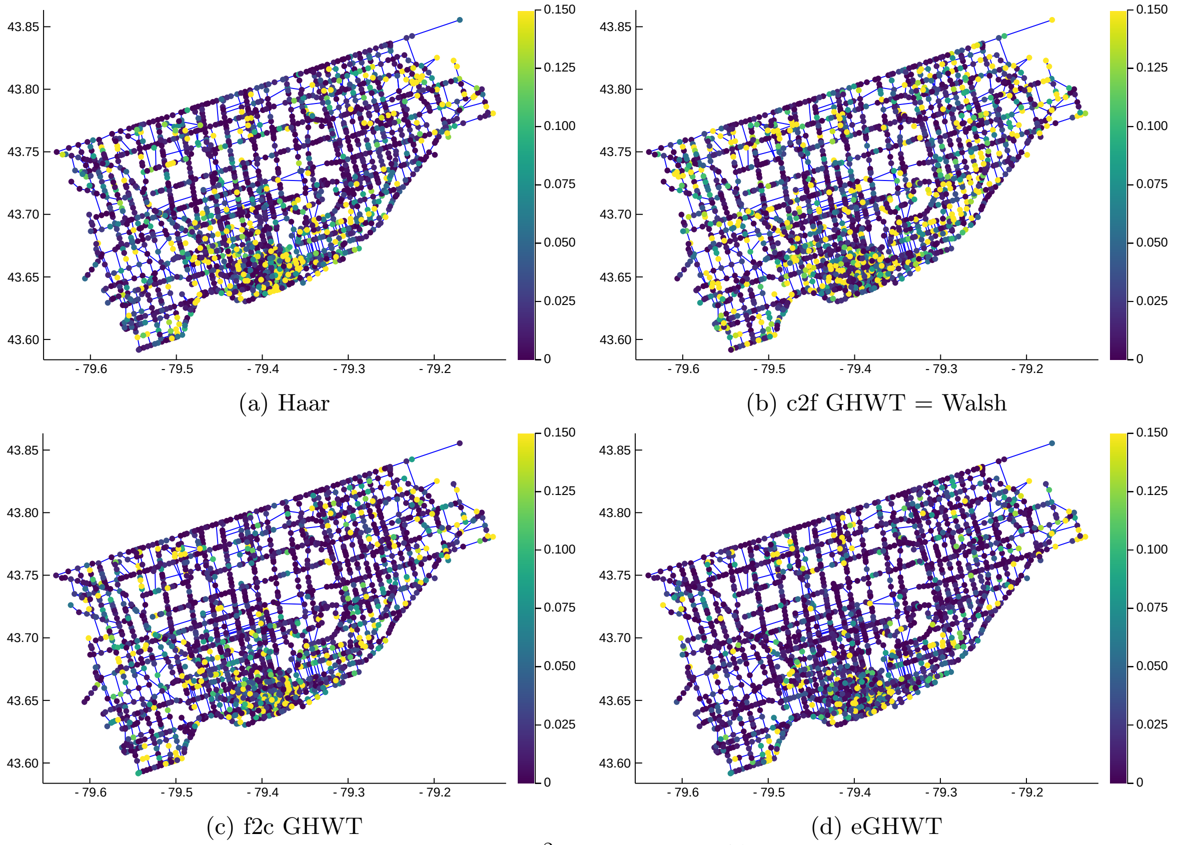


(a)



(b)

Figure 4: (a): Traffic volume data in the city of Toronto; (b): Relative $\ell^2$ approximation error of the data shown in (a)



(a) Haar

(b) c2f GHWT = Walsh

(c) f2c GHWT

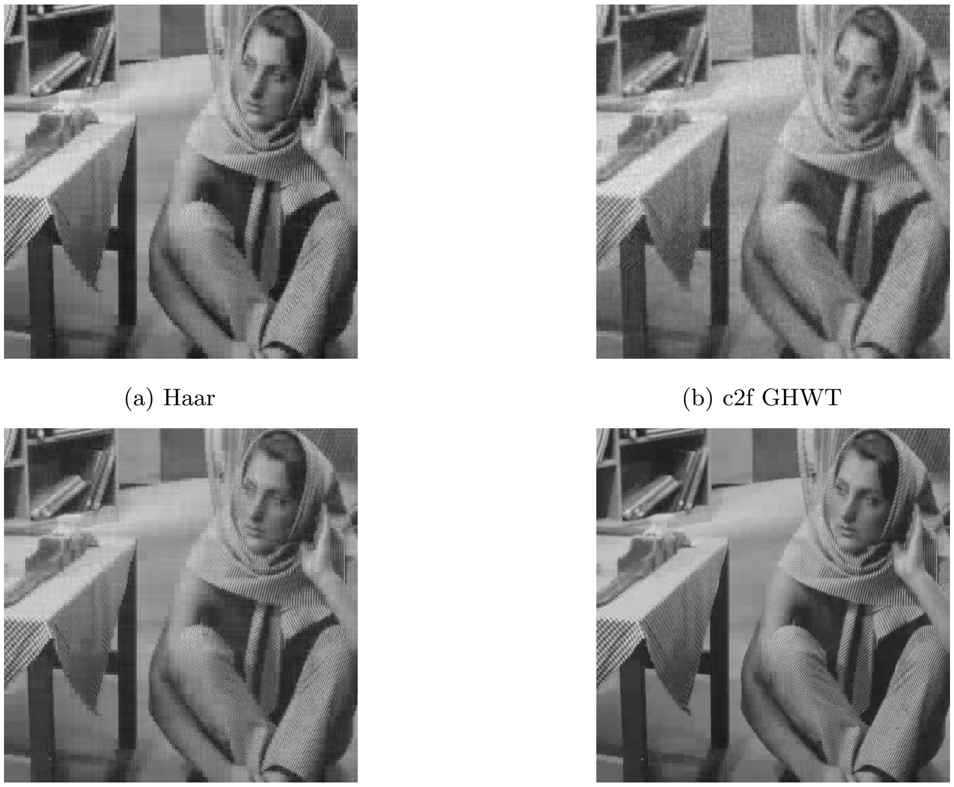
(d) eGHWT

Figure 5: Pointwise relative $\ell^2$-error using 25% of the best-basis coefficients

approximation of $\boldsymbol{f}$ with top $n$ basis vectors having largest coefficients in magnitude. The $x$-axis denotes $n/N$, i.e., the fraction of coefficients retained. In Figure 5, we display the residual plots when 25% coefficients are used. We can clearly see that the eGHWT plot is much closer to 0 than the rest of the methods.

## 4.2 Viewing an Image (or a General Matrix Signal) as a Tensor Product of Graphs

For a single signal supported on a graph, we can now use eGHWT to produce a suitable ONB. Then for a collection of signals in a matrix form (including regular digital images), we can also compose the affinity matrix



(a) Haar (b) c2f GHWT

(c) f2c GHWT (d) eGHWT

Figure 6: Comparison of various bases: using only 3.125% of coefficients

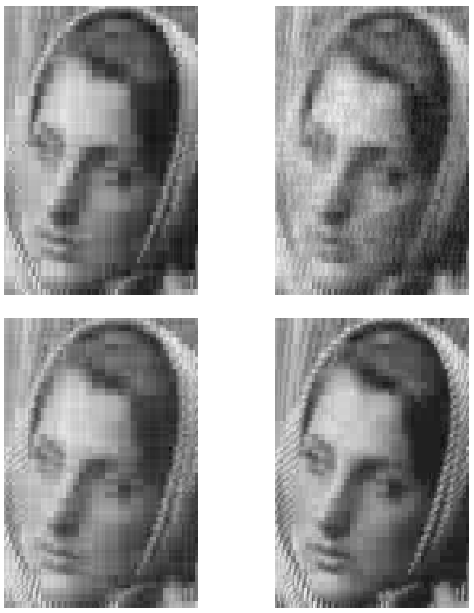
of the rows and that of the columns separately, thus define graphs that the rows and columns are supported on as was done previously.[13,25] Those affinity matrices can be either computed from the similarity of rows or columns directly or can be composed from information outside the original matrix signal. For example, Kalofolias[26] used row and column graphs to analyze recommender systems.

After the affinity graphs on rows and columns are obtained, we can use the eGHWT to produce ONBs on rows and columns separately. Then the matrix signal can be analyzed or compressed by the tensor product of those two ONBs. In addition, the Thiele-Villemoes algorithm,[20] from which we develop the eGHWT, has been extended to matrix signals by Lindberg and Villemoes.[24] We have also extended the eGHWT to the tensor product of row and column affinity graphs and search for best 2D ONB on the matrix signal directly. Note that we can also specify or compute the binary partition trees in a non-adaptive manner (e.g., recursively splitting at the middle of each region), typically for signals supported on a regular lattice.

### 4.2.1 Approximation of the Barbara Image

In this section, we use the famous Barbara image. The size of the image is $512 \times 512$. The partition trees on the rows and columns are specified explicitly: every bipartition is forced at the middle of each region. Therefore, those two trees are balanced binary trees with depth equal to $\log_2(512) + 1 = 8$.

Figure 6 displays the approximation performance of basis vectors of Haar, c2f GHWT, f2c GHWT, and eGHWT. We can see that with the same fraction ($1/32 = 3.125\%$) of the coefficients retained, the eGHWT has much higher performance with less blocky artifacts than that of Haar and c2f/2c GHWTs. Figure 7 shows the zoomed-up face and left leg of those approximations. Especially for the leg region that has some specific texture, i.e., stripe patterns, the eGHWT outperformed the rest of the methods.


(a) face

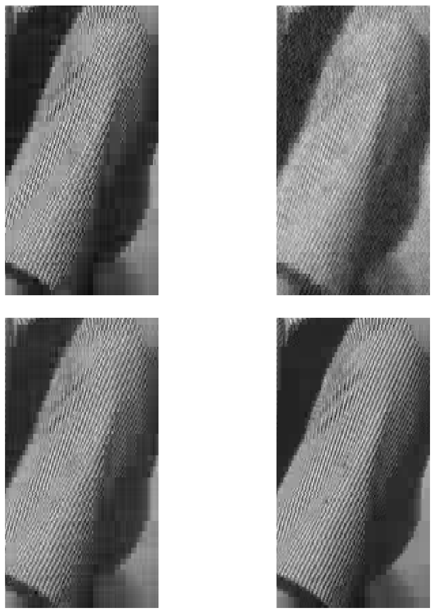

(b) leg

Figure 7: Comparison of various bases: using only 3.125% of coefficients. Methods used from left to right and top to bottom: Haar; c2 GHWT; f2c GHWT; eGHWT.

### 4.2.2 Haar Transform for Images with Non-Dyadic Size

For non-dyadic images, there is no straightforward way to obtain the partition trees in a non-adaptive manner as we did for the Barbara image in the previous subsection. This is a problem faced by the classical Haar transform as well, which requires an input image to be dyadic. Non-dyadic images are often modified by zero padding, even extension, or other methods before the Haar transform is applied. We propose to apply the Haar transform on a non-dyadic image without modifying the input image using the eGHWT.

To obtain the binary partition trees, we need to cut an input image $\boldsymbol{F}$ horizontally or vertically into two parts recursively. Apart from using the affinity matrices, we propose the *penalized total variation cost* to cut the input image. Denote the two sub-parts as $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$. We search for the optimal cut such that

$$\text{Penalized Total Variation Cost} := \frac{\|\boldsymbol{F}_1\|_{\mathrm{TV}}}{|\boldsymbol{F}_1|^p} + \frac{\|\boldsymbol{F}_2\|_{\mathrm{TV}}}{|\boldsymbol{F}_2|^p} \ (p > 0)$$

is minimized, where $\|\boldsymbol{F}_k\|_{\mathrm{TV}} := \sum_{i,j}(|F_k[i+1,j] - F_k[i,j]| + |F_k[i,j+1] - F_k[i,j]|)$, and $|F_k|$ indicates the number of pixels in $\boldsymbol{F}_k$, $k = 1, 2$. The denominator is used to make sure that the size of $\boldsymbol{F}_1$ and that of $\boldsymbol{F}_2$ are close so that the tree becomes nearly balanced. Recursively applying the horizontal cut on the rows of $\boldsymbol{F}$ and the vertical cut on the columns of $\boldsymbol{F}$ will give us two binary partition trees. We can then select the 2D Haar basis from the eGHWT dictionary or search for the best basis with minimal cost (note that this cost function for the best-basis search is different from the penalized total variation cost above).

Here we chose an image patch of size $100 \times 100$ around the face part from the original $512 \times 512$ Barbara image so that it is non-dyadic.

Here $p = 3.0$ is chosen. To decide the value of $p$, we need to balance between the total variation and structure of partition tree. Larger $p$ means less total variation value after split but less balanced partition tree. $p$ can be fine-tuned based on evaluation of the final task, for example, the area under the curve of the relative approximation error in the compression task.

Figure 8 shows that the decay speed of the eGHWT basis vector coefficients is faster than the classical Haar transform (non-adaptive cuts after even reflection to make the input image dyadic).

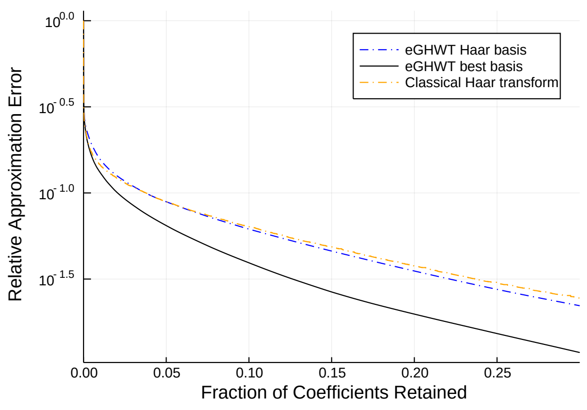


Figure 8: Comparison of the classical Haar transform and the eGHWT bases on the face part (of size $100 \times 100$) of the Barbara image

### 4.3 Constructing a Graph from an Image for Efficient Approximation

We can view a digital image of size $M \times N$ as a signal on a graph consisting of $MN$ nodes by viewing each pixel as a node. Note that the underlying graph is not a regular 2D lattice of size $M \times N$. Rather it is a graph reflecting the relationship or affinity between pixels. In other words, $w_{ij}$, the weight of the edge between $i$th and $j$th pixels in that graph should reflect the affinity between local region around these two pixels, and this weight may not be 0 even if $i$th and $j$th pixels are remotely located. This idea have been used in image segmentation[27] as well as image denoising.[28]

Here we define the edge weight $w_{ij}$ as Szlam at al.:[28]

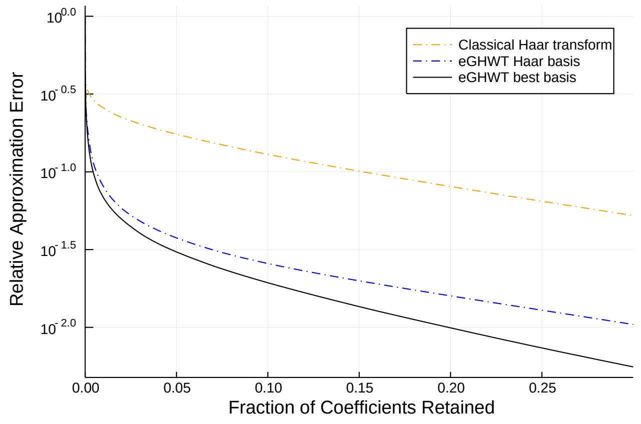
$$w_{ij} = \mathrm{e}^{\frac{-\|F[i]-F[j]\|_2^2}{\sigma_F}} \cdot \begin{cases} \mathrm{e}^{\frac{-\|X[i]-X[j]\|_2^2}{\sigma_X}} & \text{if } \|X[i]-X[j]\|_2 < r \\ 0 & \text{otherwise} \end{cases}$$

where $X[i]$ is the spatial location of node (pixel) $i$, and $F[i]$ is a feature vector based on intensity, color, or texture information of the local region centered at that node. As one can see from the above weight, the pixels located within a disk with center $X[i]$ and radius $r$ are considered to be the neighbors of the $i$th pixel. The scale parameters, $\sigma_F$ and $\sigma_X$ must be chosen appropriately. Once we construct this graph, we can apply the eGHWT in a straightforward manner.

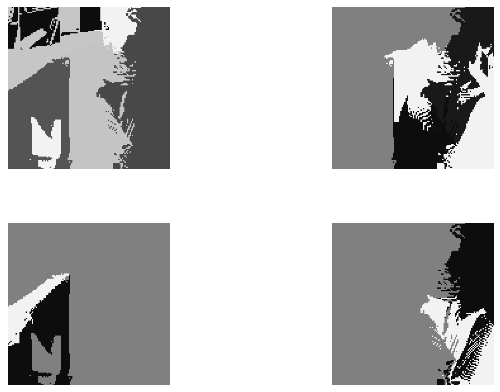
We examined two images here. The first one is the subsampled version of the Barbara image of size $128 \times 128$ (we subsampled the original Barbara image to reduce computational cost). For both experiments, we used $r = 5, \sigma_X = \infty$ while $\sigma_F = 0.001$ for the first example and 0.0001 for the second example. The parameters can firstly be chosen according to the performance of the image cut reviewed by human, then fine-tuned based on the on evaluation of the final task, for example, the area under the curve of the relative approximation error here. Also, as the feature vector $F[i]$ at the $i$th pixel, we simply used its pixel (intensity) value in those examples. We will investigate how the use of more sophisticated feature vectors would improve the image approximation performance and report our findings at a later date.

Figure 9 shows our results on the subsampled Barbara image. Figure 9a demonstrates that the decay rate of the expansion coefficients using the eGHWT dictionary is much faster than that of the classical Haar transform. Moreover, the basis vectors from the eGHWT best basis display some meaningful feature of the image. Figure 9b shows the four best-basis vectors corresponding to the largest expansion coefficients in magnitude (other than the scaling vector). We can see that the basis vectors shown in top left and bottom left interact with the table and the bookshelf in the image while the basis vector shown in top right indicates Barbara herself and the on shown in bottom right indicates her left leg region.

The second example is a composite texture image. Our method, i.e., applying the eGHWT on an image viewed as a graph, allows us to generate basis vectors with irregular support that is adapted to the structure

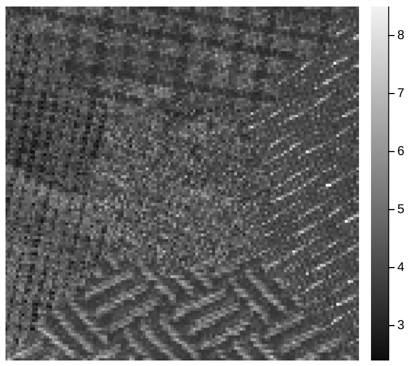


(a) Relative $\ell^2$ errors vs the sorted expansion coefficients of three bases

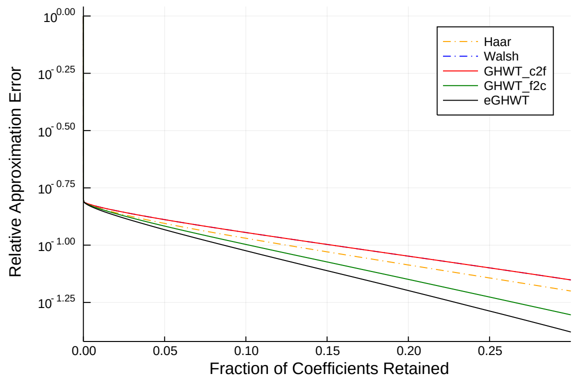


(b) Four eGHWT best-basis vectors with the largest coefficients in magnitude (excluding the scaling vector)

Figure 9: The eGHWT applied on the Barbara image viewed as a graph



(a)



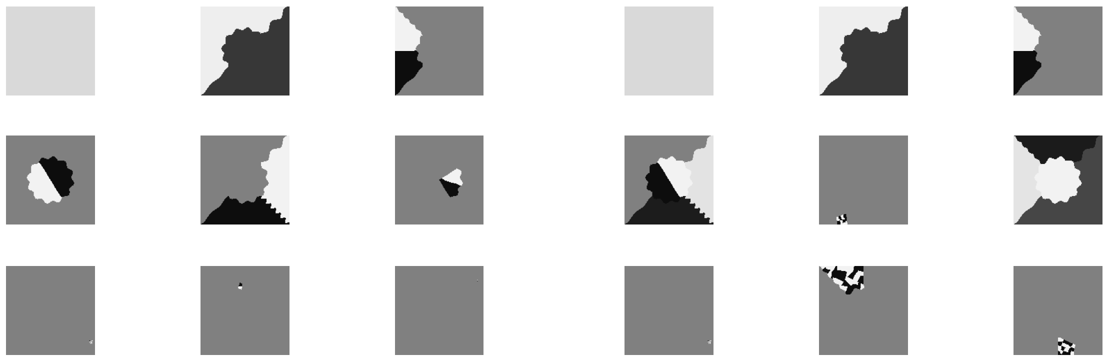
(b)

Figure 10: (a): A composite textured image of size $128 \times 128$; (b): Relative $\ell^2$ approximation error of (a) using five methods

of the input image as shown in the Barbara image example above, which works particularly well on texture images. Figure 10a displays the original composite texture image. For visualization purpose, the graph weights are computed from the template image reflecting the ground truth of the five textured regions. Figure 10b shows the approximation performance of five different methods. Figure 11 compares the top basis vectors of the graph Haar basis and those of the eGHWT best basis.

## 5. SUMMARY

In this article, we have introduced the extended Generalized Haar-Walsh Transform (eGHWT). After briefly reviewing the preceding Generalized Haar-Walsh transform (GHWT), we have described how the GHWT can be improved with the new best-basis algorithm. We call this whole procedure of developing the extended Haar-Walsh wavelet packet dictionary on a graph and selecting the best basis from it as the eGHWT. We then have conducted several numerical experiments using both synthetic and real examples to demonstrate the improvement of the eGHWT over the GHWT. Moreover, we have developed the 2D eGHWT for matrix signals, especially digital images, which has also shown superiority over the classical Haar-Walsh wavelet packet transforms. Currently, we are conducting more numerical experiments of 2D eGHWT, including the analysis of term-document matrices and recommender system. We hope to report these at a later date.



(a) (b)

Figure 11: (a): Top 9 Haar basis vectors computed from the GHWT f2c dictionary; (b): Top 9 eGHWT best basis vectors

## ACKNOWLEDGMENTS

This research was supported in part by N.S.'s NSF grants DMS-1418779 and DMS-1912747. In addition, Y.S. was supported by 2017–18 Summer Graduate Student Researcher Award by the UC Davis Office of Graduate Studies. The authors thank Haotian Li of UC Davis for generating the graph of the Toronto street map.

## REFERENCES

[1] Chung, F. and Lu, L., [*Complex Graphs and Networks*], no. 107 in CBMS Regional Conference Series in Mathematics, Amer. Math. Soc., Providence, RI (2006).

[2] Easley, D. and Kleinberg, J., [*Networks, Crowds, and Markets: Reasoning and a Highly Connected World*], Cambdrige Univ. Press, New York (2010).

[3] Lovász, L., [*Large Networks and Graph Limits*], vol. 60 of *Colloquium Publications*, Amer. Math. Soc., Providence, RI (2012).

[4] Newman, M., [*Networks*], Oxford Univ. Press, 2nd ed. (2018).

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

[6] Ortega, A., Frossard, P., Kovačević, J., Moura, J. M., and Vandergheynst, P., "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE* **106**(5), 808–828 (2018).

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

[8] Coifman, R. R. and Maggioni, M., "Diffusion wavelets," *Appl. Comput. Harm. Anal.* **21**(1), 53–94 (2006).

[9] Bremer, J. C., Coifman, R. R., Maggioni, M., and Szlam, A. D., "Diffusion wavelet packets," *Appl. Comput. Harm. Anal.* **21**(1), 95–112 (2006).

[10] Murtagh, F., "The haar wavelet transform of a dendrogram," *J. Classification* **24**(1), 3–32 (2007).

[11] Lee, A., Nadler, B., and Wasserman, L., "Treelets—an adaptive multi-scale basis for sparse unordered data," *Ann. Appl. Stat.* **2**, 435–471 (2008).

[12] Jansen, M., Nason, G. P., and Silverman, B. W., "Multiscale methods for data on graphs and irregular multidimensional situations," *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **71**(1), 97–125 (2009).

[13] Coifman, R. R. and Gavish, M., "Harmonic analysis of digital data bases," in [*Wavelets and Multiscale Analysis: Theory and Applications*], Cohen, J. and Zayed, A. I., eds., *Applied and Numerical Harmonic Analysis*, 161–197, Birkhäuser, Boston, MA (2011).

[14] Hammond, D. K., Vandergheynst, P., and Gribonval, R., "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harm. Anal.* **30**(2), 129–150 (2011).

[15] Rustamov, R. M., "Average interpolating wavelets on point clouds and graphs," *arXiv preprint arXiv:1110.2227* (2011).

[16] Irion, J. and Saito, N., "The generalized Haar-Walsh transform," in [*Proc. 2014 IEEE Workshop on Statistical Signal Processing (SSP)*], 472–475, IEEE (2014).

[17] Irion, J. and Saito, N., "Applied and computational harmonic analysis on graphs and networks," in [*Wavelets and Sparsity XVI, Proc. SPIE 9597*], Papadakis, M., Goyal, V. K., and Van De Ville, D., eds. (2015). Paper # 95971F.

[18] Irion, J. and Saito, N., "Efficient approximation and denoising of graph signals using the multiscale basis dictionaries," *IEEE Trans. Signal Inform. Process. Netw.* **3**(3), 607–616 (2017).

[19] Coifman, R. R. and Wickerhauser, M. V., "Entropy-based algorithms for best basis selection," *IEEE Trans. Inform. Theory* **38**(2), 713–718 (1992).

[20] Thiele, C. M. and Villemoes, L. F., "A fast algorithm for adapted time–frequency tilings," *Appl. Comput. Harm. Anal.* **3**(2), 91–99 (1996).

[21] Von Luxburg, U., "A tutorial on spectral clustering," *Stat. Comput.* **17**(4), 395–416 (2007).

[22] Fiedler, M., "A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory," *Czechoslovak Math. J.* **25**(4), 619–633 (1975).

[23] Saito, N. and Coifman, R. R., "Extraction of geological information from acoustic well-logging waveforms using time-frequency wavelets," *Geophysics* **62**(6), 1921–1930 (1997).

[24] Lindberg, M. and Villemoes, L. F., "Image compression with adaptive haar-walsh tilings," in [*Wavelet Applications in Signal and Image Processing VIII, Proc. SPIE 4119*], 911–922, International Society for Optics and Photonics (2000).

[25] Irion, J. and Saito, N., "Learning sparsity and structure of matrices with multiscale graph basis dictionaries," in [*Proc. 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*], Uncini, A., Diamantaras, K., Palmieri, F. A. N., and Larsen, J., eds. (2016).

[26] Kalofolias, V., Bresson, X., Bronstein, M., and Vandergheynst, P., "Matrix completion on graphs," *arXiv preprint arXiv:1408.1717* (2014).

[27] Shi, J. and Malik, J., "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Machine Intell.* **22**(8), 888–905 (2000).

[28] Szlam, A. D., Maggioni, M., and Coifman, R. R., "Regularization on graphs with function-adapted diffusion processes," *J. Mach. Learn. Res.* **9**(Aug), 1711–1739 (2008).