Multidimensional Systems and Signal Processing (2026) 37:7
https://doi.org/10.1007/s11045-026-00908-1

Efficient multidimensional signal decorrelation through Cayley graphs

Aviraj Sinha[1] · Darrell L. Young[1] · Eric C. Larson[1] · Mitchell A. Thornton[1]

Received: 30 November 2024 / Revised: 30 November 2024 / Accepted: 4 August 2026

Abstract
In this work, we generate Cayley graphs to obtain spectra corresponding to orthogonal eigenvectors that efficiently decorrelate multidimensional signals. Optimal signal decorrelation information is commonly obtained from the Karhunen-Loeve transform. However, whereas the traditional KL-transform autocovariance matrix is generated using multiple products between signal components, generating the Cayley graph only requires memory access operations for assigning weights to the graph edges. The graph is generated by coloring the edges between different vertices, where the vertices represent symmetry group elements, and the weight values are accessed from the discrete multidimensional functions. The efficiency from avoiding product calculations allows the use of spectral information for a wide variety of applications, as the sparsity of the spectra indicates the degree to which a function can be maximally decorrelated. For real-world applications, we begin by demonstrating how Cayley graph spectra can be used for one-dimensional signal classification. We extend our theory to multiple dimensions, allowing us to develop higher-dimensional applications. We obtain the Cayley graph representation of multiple two-dimensional image patches, which demonstrates that the spectra at each window patch can be used to match key points between transformed images. As another multidimensional example, we apply the method to a three-dimensional image patch; this is then used for boundary edge detection of three-dimensional scans. Thus, our method is useful for extracting valuable decorrelation information from signals of any number of dimensions, and it can outperform common signal processing techniques for a diverse set of tasks.

Keywords Graph signal processing · KL transform · Graph spectra · Discrete function

✉ Aviraj Sinha
avirajs@smu.edu

Darrell L. Young
dlyoung@smu.edu

Eric C. Larson
eclarson@smu.edu

Mitchell A. Thornton
mitch@lyle.smu.edu

[1] Darwin Deason Institute of Cybersecurity, Southern Methodist University, 6425 Boaz Lane, Dallas, TX 75205, USA

Published online: 16 August 2026

1 Introduction

The Karhunen-Loeve transform is one of the most influential algorithms in the field of signal processing and data processing (Karhunen, 1946). The goal of the KL transform is to decorrelate or decompose a function into a set of linearly uncorrelated eigenfunctions, often referred to as principal components. The corresponding eigenvalues in the spectra are related to how much each principal component contributes to reconstructing the original signal. The eigenvectors are used to construct the transform matrix, which, when applied, maximizes the number of zero values of the transformed signal. The KL transform plays a significant role in machine learning, frequently applied for both dimensionality reduction (Liu et al., 2017) and feature selection (Kittler and Young, 1973), highlighting its value in data analysis. This ability of the KL transform to find the optimal basis functions that highlight important features enables its application for various signal processing tasks. This includes applications such as multichannel noise reduction (Lacouture-Parodi et al., 2014), image noise suppression (Al-Asad et al., 2009), and signal compression (Olmos et al., 1996).

Applying the KL transform first requires creating the matrix known as the autocovariance matrix, which consists of multiplications between components: $\mathbf{R_x} = Cov\{\mathbf{x}\} = E\{\mathbf{x}\mathbf{x}^T\}$. The eigenvectors v_i of these components are then used to create the KL transform matrix $\mathbf{U}$. The KL matrix is then applied to the original signal, which results in maximal compaction or, in other words, the maximum number of zero values in the response signal: $\mathbf{Y} = \mathbf{U}^T(\mathbf{X} - \mu)$. Since both the KL transform and the autocovariance matrix share the same eigenvectors, the KL spectrum is used to refer to the eigenvalues of the autocovariance matrix. The KL spectra are related to the signal's ability to be decorrelated using a few eigenvectors, which is why the KL spectra are sparse. Forming a selection of optimal basis functions that decorrelate the signal is possible since the transform matrix represents the statistical information of the data and is constructed using the data itself. In Sect. 2.2 we discuss the generation of the KLT in-depth and its relationship to our transformation.

In this work, we use the Cayley graph representation of a signal and find the eigenvalues of the symmetric Cayley graph adjacency matrix to extract the decorrelated spectral information, as illustrated in Fig. 1. The constructed Cayley graph adjacency matrix, described in Sect. 4.2, is a symmetric matrix that produces orthogonal eigenvectors that maximize the number of zero real-valued eigenvalues. As shown in Fig. 1, generating the Cayley graph involves using the proper generator function that extracts the decorrelation information; this generator function is related to the permuted columns of the autocorrelation matrix since the elements of the Cayley graph are that of the symmetry permutation group. The multidimensional autocorrelation definitions and notation are discussed in depth in Sect. 2.1. The

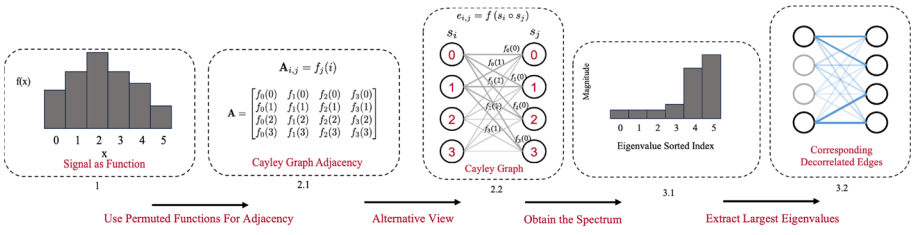


Fig. 1 Process of Cayley graph decorrelation illustrated through steps: **1** signal representation as a function, **2.1** construction of the Cayley graph adjacency matrix using permuted functions, **2.2** visualization of the Cayley graph structure, **3.1** extraction of eigenvalues sorted by magnitude, and **3.2** identification of decorrelated edges based on the largest eigenvalues

Cayley graph represents the relationship of different signal components with connected edge weights representing their permuted values. As a result, rather than obtaining decorrelation information through product calculations, we are representing the impact of various permutations on each component value. In this graph structure, the node clusters with similar values signify the permuted signal components are low variance, resulting in greater decorrelation and more zero values in the spectra since fewer principal components are needed to represent them. This Cayley graph definition that contains the symmetry permutation group elements aligns with the method used to generate similar spectra related to the KL spectra from one-dimensional signals in Thornton (2005). We discuss our related work in Sect. 3.2 that similarly uses Cayley graphs created from relevant generator functions to extract the spectra related to other transforms.

We first describe the process to generate the spectra of the Cayley graph that are related to the multidimensional KL spectra in Sect. 4.2. In Sect. 5, we validate the decorrelation ability of the Cayley graph by analyzing the spectral distribution and sparsity. We also show that the Cayley graph spectra can be calculated more efficiently than the KL spectra by comparing timing information. In addition, we analyze how the mapping of signal indices is an important factor in the decorrelation of a signal since the permutations may have less of an impact on certain mappings where similar signal components are nearby. In Sect. 6, we then explore the application of Cayley graphs to obtain higher-dimensional decorrelation information and include examples of how to use the spectra and decorrelation information for signal and data processing. First, for the 1D applications, signal classification for both audio and ECG can be done using our Cayley graph method. Then, we utilize our technique for higher-dimensional signals by running our method across small image patches, the spectra being used as extracted feature vectors. These feature vectors are then used to match portions of transformed images. The image feature matching results in more matches obtained than the most popular current methods while using smaller feature vectors. Finally, we show that our work allows Cayley graphs to represent higher-dimensional 3D images and that there are performance improvements in viewing details for boundary detection.

2 Signal processing background

2.1 Autocorrelation matrices

An autocorrelation matrix would allow us to analyze the repetition of different portions of the signal, which, when used to generate the Cayley graph representation, can then be used to decorrelate the signal into orthogonal basis functions. To define the autocorrelation matrix, we first start by introducing the autocorrelation function notation as described by Thornton (2005). The autocorrelation function is the measure of a function's correlation with itself. Note that the correlation is simply a convolution of one signal with a time-reversed version of itself. The autocorrelation function can then be described as an equation $R(u) = \sum_{x=0}^{N-1} f(x) f(x \ominus_n u)$; this contains the correlation of a 1D signal on itself with all positive lags u and N being the length of the signal. This function definition can be expanded to work on discrete multidimensional signals. The method to obtain the 2D discrete autocorrelation function $R(u, v)$ for every pixel response at (u, v) is shown in Eq. 1. Here, m and n are the dimensions of the original 2D image $\mathbf{I}$. The function $I(x, y)$ represents the pixel values of the original 2D image at coordinates (x, y), and $\ominus_m$ and $\ominus_n$ represent the modulo subtraction operation with respect to the dimensions m or n, respectively. This formulation

simply represents the correlation of the image $\mathbf{I}$ with itself. This can be extended to any number of dimensions as needed.

$$R(u, v) = \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} I(x, y) \cdot I\left((x \ominus_m u), (y \ominus_n v)\right) \tag{1}$$

In our work, we use the autocorrelation matrix, which, when multiplied with a flattened image, results in the autocorrelation function output described in Eq. 1. We represent the discrete autocorrelation matrix and the operation on the original signal as a matrix–vector multiplication: $\mathbf{R_u} = \mathbf{FI}$. Here $\mathbf{F}$ is the autocorrelation matrix, $\mathbf{I}$ is the original function or flattened image if two-dimensional, and $\mathbf{R_u}$ is the autocorrelation function as a flattened vector of responses. Each column of the autocorrelation matrix is a different permutation of the original data value. In the case of a 1D signal, the columns consist of the signal at different lags, and for a 2D image, the columns are shifted along both axes for different amounts before flattening.

To help construct the autocorrelation matrices for 2D and n-D signals, we can first denote the 2D image function $\mathbf{I_f}$, which can also be thought of as the image $\mathbf{I}$ when flattened or stacked along the x-axis or taking each row of a matrix and concatenating them side-by-side. We can also represent sliding the image $\mathbf{I_f}(\mathbf{x} \ominus \mathbf{i}, \mathbf{y} \ominus \mathbf{j})$ across either or both of the x and y axes before flattening the resulting 2D matrices into columns, $\mathbf{I_f}(\mathbf{x}, \mathbf{y})$, where i and j are values used to slide the image by i pixels along the x-axis and j pixels along the y-axis before flattening. The sliding permutations of the matrix are denoted by the $\ominus$, corresponding to a subtraction-modulo operator, to show the shifting of the image. We can then arrange the various $\mathbf{I_f}(\cdot)$ values for different values of i, j to construct an autocorrelation matrix $\mathbf{F}$ as shown in Eq. 2. Note that each column, when multiplied with the original $\mathbf{I_f}$, results in a specific response component of $\mathbf{R_u}$.

$$\mathbf{F}_{\ominus} = [\mathbf{I_f}(\mathbf{x} \ominus \mathbf{0}, \mathbf{y} \ominus \mathbf{0}) \ldots \mathbf{I_f}(\mathbf{x} \ominus (\mathbf{N} - 1), \mathbf{y} \ominus (\mathbf{M} - 1))] \tag{2}$$

Programmatically, we can construct the matrix $\mathbf{F}$ by sliding the image across the x, y axes one pixel at a time from left to right, then move it down, and finally flatten and arrange the columns into the autocorrelation matrix. Extending the process to N-D signals involves a similar function, but instead of sliding the function $\mathbf{I_f^{x,y}}$ across two axes x, y, we can slide it across any number of axes $x_0, x_1, \ldots x_i$. We can then flatten each resulting N-D matrix into a row vector and arrange these vectors to form the desired circulant matrix. We denote the N-D image function $\mathbf{I_f}$ as $\mathbf{I_f}(x_0 \ominus N_0, \ldots, x_n \ominus N_n)$, where $N_0 \ldots N_n$ are the shift values for those axes. The autocorrelation matrix $\mathbf{F}$ will have dimensions $x_0 \times x_1 \ldots \times x_n$. Now, to create the N-D signal's autocorrelation matrix $\mathbf{R}$, we slide $\mathbf{I}$ across all axes for different amounts. Then, we flatten the resulting matrix into a column vector. After that, we arrange these column vectors into the desired autocorrelation matrix $\mathbf{R}$. Then, the autocorrelation matrix $\mathbf{F}$ can be constructed as shown in Eq. 3.

$$\begin{aligned} \mathbf{F}_{\ominus} = \Big[& \mathbf{I_f}\left(\mathbf{x_1} \ominus \mathbf{0}, \mathbf{x_2} \ominus \mathbf{0}, \ldots, \mathbf{x_n} \ominus \mathbf{0}\right), \\ & \ldots, \\ & \mathbf{I_f}\left(\mathbf{x_1} \ominus N_1, \mathbf{x_2} \ominus N_2, \ldots, \mathbf{x_n} \ominus N_n\right) \Big] \end{aligned} \tag{3}$$

The **F** autocorrelation matrix, created from a one-dimensional signal, is a circulant matrix where each row is a cyclic shift of the previous row. A double-block (or N-block for N dimensions) circulant matrix extends the autocorrelation matrix generation for signals of multiple dimensions. Each block in a block-circulant matrix is a circulant matrix, which is repeated by descending along the diagonal. The description of an autocorrelation matrix as a data-dependent representation with valid permutations of the signal in each row is described in Thornton (2005). This description of the autocorrelation matrix through permutations is then extended in our Cayley graph representation. We can motivate the need for the Cayley graph representation to extract more optimized spectral information through more complex permutations since we show that circulant matrices only effectively decorrelate sinusoidal signals and the fact that the N-block circulant autocorrelation is not generally symmetric and has complex-valued spectra.

2.2 Decorrelated spectral transforms

Our work focuses on efficiently finding decorrelated and sparse spectra, similar to the KL spectra, while avoiding multiplications. We emphasize the important properties of the KL spectra by comparing the KL transformation matrix to the discrete Fourier transform (DFT) matrix. As we will demonstrate, the KLT results in optimal decorrelation, and as the data becomes periodic, the KLT becomes the DFT. This motivates the generation of a Cayley graph, which can be used to obtain signal decorrelation information without requiring the product operations to generate the autocovariance matrix. Like the autocovariance matrix, our Cayley graph results in a symmetric adjacency matrix with orthogonal eigenvectors that are more optimized to capture variance information than the DFT. We further outline the properties of the Cayley graph that give us decorrelated spectral information in this section.

First, we discuss the DFT to motivate the usage and optimal decorrelation of the KL spectra. The discrete Fourier transform (DFT) can be represented by a matrix known as the DFT matrix, as shown in Eq. 4. Here, the transfer matrix is created by knowing its size n, and the computation is made from the sinusoidal basis vectors at each column. $\mathbf{T}_{DFT}$ is defined in Eq. 4. When this transfer matrix is applied to $\mathbf{x}$, the resulting $\mathbf{y}$ consists of Fourier components presenting the frequency of the original data.

$$\mathbf{T}_{DFT} = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \dots & \omega^{(n-1)^2} \end{bmatrix} \tag{4}$$

The KL transform described as $y = \mathbf{T}_{KL}\mathbf{x}$, applies the transformation matrix $\mathbf{T}_{KL}$ on the signal $\mathbf{x}$, where the components are $x_k \in \mathrm{R}$. The $\mathbf{y}$ is the transformed vector to a new coordinate space where the components are uncorrelated; this results in the variation primarily being captured only using the first few components of $\mathbf{y}$, allowing for data compression by discarding all but few of the $\mathbf{y}$ components. To create the transform, we assume $\mathbf{x}$ is a normalized zero-mean random vector $E\{\mathbf{x}\} = 0$, obtained from the operation $\mathbf{x} - \mu_{\mathbf{x}}$. The transform matrix $\mathbf{T}_{KL}$ is obtained from the eigenvectors of the autocovariance matrix of $\mathbf{x}$. As we discussed, the eigenvectors are ordered and used to construct the $\mathbf{T}_{KL}$. For this reason, the eigenvalues of the autocovariance matrix and the KL transform can be described as either the KL spectra or autocovariance spectra. We then need to find the transformation matrix,

$\mathbf{T}_{KL}$, which can be created from the eigenvalues of the autocovariance matrix. We can denote the basis vectors t_i and corresponding eigenvalues here $\lambda_1, \lambda_2, ..., \lambda_N$. The transformation matrix is created here where the eigenvectors are ordered by the eigenvalues in decline order $\mathbf{T}_{KL} = [t_1, t_2, ..., t_N]$ Thus, the eigenvectors corresponding to large eigenvalues will represent the principal components along which most variation exists, and the eigenvalues are the amount of variation captured by those principal components. A result is that we can take the first M component and that the coefficients of $y_k = 0$ for all $M < k < N - 1$. $y = [y_0, y_1 ... y_{M-1} ... 0..0]$. This means we would only need to take the subset of the eigenvectors of t_0 up to t_M since the rest of the transformation matrix rows are irrelevant. Since the overall magnitudes of the $||y_m||^2$ are still the same as $||x||^2$, this subset of eigenvectors can then be used to maximally compact the information to fewer signal components.

The autocovariance matrix is defined from the $\mathbf{R_x} = Cov\{\mathbf{x}\} = E\{\mathbf{x}\mathbf{x}^T\}$. An example matrix is shown in Eq. 5. In the equation, we show that if the x variable is a 1D signal, the components are simply multiplied since the inner product of scalars is multiplication. Extending the autocovariance definition to higher dimensions would require an increasingly higher number of multiplications for each dimension since the variable x can also represent higher dimensional signals when the signal is flattened. This flattening operation is similar to the process described in generating the autocorrelation matrix columns in Sect. 2.1. If, for example, each dimension has the same size n, the size of the flattened vector would scale in size n^d, where d is the total number of dimensions. This would lead to an increasingly high number of product calculations. Note that this $\mathbf{R_x}$ forms a symmetric matrix since one of the properties of inner products is that they are symmetric: $\langle x_a, x_b \rangle = \langle x_b, x_a \rangle$.

$$\begin{aligned} \mathbf{R}_x &= \begin{bmatrix} \langle x_0, x_0 \rangle & \langle x_0, x_1 \rangle & \cdots & \langle x_0, x_{N-1} \rangle \\ \langle x_1, x_0 \rangle & \langle x_1, x_1 \rangle & \cdots & \langle x_1, x_{N-1} \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle x_{N-1}, x_0 \rangle & \langle x_{N-1}, x_1 \rangle & \cdots & \langle x_{N-1}, x_{N-1} \rangle \end{bmatrix} \\ &= \begin{bmatrix} x_0^2 & x_0 x_1 & \cdots & x_0 x_{N-1} \\ x_1 x_0 & x_1^2 & \cdots & x_1 x_{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N-1} x_0 & x_{N-1} x_1 & \cdots & x_{N-1}^2 \end{bmatrix} \end{aligned} \tag{5}$$

We now show that the previous autocovariance in Eq. 5 becomes a circulant matrix when created from periodic signals. As we discussed earlier, the previous autocorrelation definition in Eq. 2 is also a circulant matrix, so this relation may give some insight. An increase in the periodic nature of the KLT eigenvectors can first be shown that the autocovariance, which takes the form of a symmetric matrix, also starts becoming a circulant matrix as the data becomes periodic, such as the signal $x_k = x_{k+m}$, where m is the period of the signal. Through the inner products of the autocovariance matrix, we know that with more periodic signals, the inner products themselves would repeat since the inner product between any two components x_i and x_j would only depend on the difference m: $\langle x_i, x_j \rangle = \langle x_{i+m}, x_{j+m} \rangle$. We can show this by reconstructing $\mathbf{R_x}$ in Eq. 6. This matrix is circulant since the values that appear at the end of the vector reappear in the front of the next row, illustrating a period of N in the original signal. The circulant structure appears in the autocovariance due to the similar responses r_i from the multiplications appearing multiple times.

$$\mathbf{R_x} = \begin{bmatrix} r_0 & r_1 & \cdots & r_{N-1} \\ r_{N-1} & r_0 & \cdots & r_{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ r_2 & r_3 & \cdots & r_1 \\ r_1 & r_2 & \cdots & r_0 \end{bmatrix} \tag{6}$$

Next, we show that circulant matrices generated from the KLT with periodic data directly relate to the DFT. For example, the circulant matrix, $\mathbf{C}$ of size $n \times n$, has eigenvalues directly related to the DFT of the original signal $\mathbf{c}$. Let $\mathbf{c} = [c_0, c_1, \ldots, c_{n-1}]$ be the first column of $\mathbf{C}$, and $\mathbf{P}$ be the permutation matrix that cyclically shifts a vector by one position. Then, $\mathbf{C}$ can be expressed as: $\mathbf{C} = \mathbf{c}_0\mathbf{I} + \mathbf{c}_1\mathbf{P} + \mathbf{c}_2\mathbf{P}^2 + \ldots + \mathbf{c}_{n-1}\mathbf{P}^{n-1}$. Thus the eigenvalues λ_k of $\mathbf{C}$ are related to the DFT of $\mathbf{c}$ through this expression: $\lambda_k = \sum_{j=0}^{n-1} \mathbf{c}_j \omega_n^{jk}$, where $\omega_n = e^{-2\pi i/n}$ is the n-th root of unity. As a result, the eigenvectors of $\mathbf{C}$ are also that of the DFT matrix $\mathbf{T}_{DFT}$. The sinusoidal basis vectors seen in the columns of Eq. 4 can be determined and are found to consist of the Fourier basis vectors, $\mathbf{w}_N^k$, used to construct the DFT transformation in Eq. 4. This relationship informs us that simply obtaining the eigenvalues of and eigenvectors of the autocorrelation matrix in Sect. 2.1 alone will not result in optimally selected eigenvectors unless the signal is periodic since circulant autocorrelation matrices alone produce only complex sinusoidal basis functions related to the DFT. Moreover, autocorrelation matrices are not guaranteed to be symmetric for higher dimensional n-block circulant matrices and will not have orthogonal eigenvectors to decorrelate the signal.

We have shown that the autocovariance matrix used to obtain the KL spectra can become a circulant matrix as the signal becomes periodic, and then we show that a circulant matrix has eigenvalues that are the DFT coefficients. As a result, we point out that the KLT becomes the DFT for highly periodic data because the KLT eigenvectors that maximize the variance also become sinusoidal, becoming the sinusoidal eigenvectors of the DFT. As a signal loses its periodic structure, the DFT becomes less optimal in terms of energy compaction. Like the KL transform matrix, the Cayley graph adjacency matrix is a data-dependent symmetric matrix that always results in real-valued eigenvalues corresponding to orthogonal eigenvectors. Similar properties to the KL spectra arise from the relationships between the signal components since, in the Cayley graph, every signal component is an element connected to its permutations. These edge connections allow us to analyze the variation of a signal in terms of the impact of permutations on each signal component. The eigenvalue decomposition of the Cayley graph adjacency matrix, as illustrated in Eq. 7, provides orthogonal basis vectors V_{CG} that are aligned to represent the repeating structures in the data, much like the principal components in PCA or the KL transform. In later sections, we also demonstrate that Cayley graph adjacency spectra also exhibit a pattern of sparsity in the eigenvalues, matching the similar behavior of the sparse values of KL spectra and revealing spectral features that of the circulant autocorrelation.

$$A_{CG} = V_{CG}\Lambda_{CG}V_{CG}^T \tag{7}$$

We begin by stating the Spectral Theorem 2.1, which provides the foundation for analyzing the spectral properties of symmetric matrices.

Theorem 2.1 (Spectral Theorem for Symmetric Matrices) *Let M be a real symmetric matrix. Then there exists an orthogonal matrix V such that*

$$M = V \Lambda V^T,$$

where Λ is a diagonal matrix containing the real eigenvalues of M, and the columns of V are orthogonal eigenvectors of M. Consequently, the eigenvalues of M are real, and its eigenvectors can be chosen to form an orthogonal basis.

To confirm the symmetry of Cayley graphs generated by abelian groups, we apply Theorem 2.2, which establishes that the adjacency matrix A_{CG} is symmetric under the condition that the generating group is abelian, meaning all group elements commute. The properties of these Cayley graphs are explored in the next section.

Theorem 2.2 (Symmetry of Cayley Graphs) *Let G be a finite abelian group with a generating set S. If S is closed under inversion, meaning that for each $g \in S$, the inverse $g^{-1} \in S$ as well, then the adjacency matrix A_{CG} of the Cayley graph generated by G and S is symmetric.*

In Proof 2.2, we demonstrate that applying the Spectral Theorem 2.1 to the adjacency matrix A_{CG} of a Cayley graph that satisfies Theorem 2.1 enables an analysis of its spectral properties in terms of its decorrelation capabilities. This is achieved through the orthogonal eigenvectors of A_{CG}, analogous to the orthogonal eigenvectors in the KL transform, which are fundamental in signal decorrelation.

Proof Let $\text{Cay}(G, N)$ be a Cayley graph generated by a symmetric set of generators N. We denote its adjacency matrix by A_{Cay}. The matrix A_{Cay} satisfies $A_{\text{Cay}} = A_{\text{Cay}}^T$ because for any $x, y \in G$, $(x, y) \in E \Rightarrow (y, x) \in E$ due to the symmetry of N. Therefore, by the Spectral Theorem, A_{Cay} has an orthogonal decomposition $A_{\text{Cay}} = V_{\text{Cay}} \Lambda_{\text{Cay}} V_{\text{Cay}}^T$ with real eigenvalues. □

Unlike simple circulant matrices, the adjacency matrix A_{CG} of a Cayley graph encodes transformation information through edges that represent data permutations. This structure results in sparse spectra by capturing patterns and correlations inaccessible through the autocorrelation matrix, which only has sinusoidal eigenvectors. Using the spectral decomposition of the Cayley graph adjacency matrix, as shown in Eq. 7, we show the Cayley graph shares characteristics with the KL transformation matrix. In this work, we characterize Cayley graph adjacency matrices as a distinct class of symmetric matrices whose spectral decomposition provides real, orthogonal eigenvectors and a sparse set of eigenvalues. This orthogonality allows the Cayley graph's eigenvectors to act as a decorrelating basis, and the sparse spectra indicate effective isolation of the underlying structures in the data, similar to the principal components in PCA. Thus, by encoding the dataset's inherent permutations, the Cayley graph spectra reveal structural patterns that complement the conventional KL spectral analysis.

3 Group theory background

3.1 Cayley graphs

Definition 3.1 A **group** $(G, *)$ has a binary operation $*$ such that for any elements $a, b, c \in G$, the operation is closed, meaning $a * b \in G$; it is associative, satisfying $(a * b) * c = a * (b * c)$;

there exists an identity element $e \in G$ for which $a * e = e * a = a$; and every element $a \in G$ has an inverse element $a^{-1} \in G$ such that $a * a^{-1} = a^{-1} * a = e$ (Gallian, 2021). A group element is any member of the set G, and the group product operator defines how elements are combined.

Previous research (Thornton, 2005) demonstrated that generating a Cayley graph whose elements are that of the symmetry permutation group $\mathbf{S_n}$ yields spectra related to the decorrelation information from the autocorrelation matrix. In this section, we explain the group structure and how Cayley graphs can be created to represent the different operations between group elements. The formal definition and properties of groups, also followed by symmetry permutation groups, are listed in Definition 3.1. The symmetry permutation group consists of elements that represent permutations of unique items. For instance, the group $\mathbf{S_3} = (S, \circ)$ can represent unique string objects $S = \{abc, bca, cab, bac, cba, acb\}$. Each string permutation can be represented numerically as array indexes by assigning values $a = 1$, $b = 2$, and $c = 3$. These six elements in S_3 are described using the notation in Eq. 8. The top row indicates the original order 1 2 3, and the bottom row represents the reordered sequence. For instance, the element labeled as 1 corresponds to permuting abc to the string cab since the top row is 1 2 3 and the bottom row is 3 1 2.

$$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix} \equiv 0 \quad \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix} \equiv 1 \quad \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix} \equiv 2$$
$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix} \equiv 3 \quad \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix} \equiv 4 \quad \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix} \equiv 5 \tag{8}$$

Using these group elements representing possible permutations, we can construct the Cayley table, where the binary operator $\circ$ combines elements, as demonstrated in Table 9. The leftmost column of Table 9 represents the left operand of $\circ$, while the top of each column represents the right operand. This table illustrates that the $\mathbf{S_3}$ operation $\circ$ is non-commutative, exemplified by the operation $2 \circ 5 = 5$ and that $5 \circ 2 = 3$ The operation $\circ$ is non-commutative, $\mathbf{S}_3$ is non-Abelian. Since the edges are created from the unique pairs of elements and the $circ$ is non-commutative, the matrix representing the directed Cayley graph is non-symmetric. We will show in another example that when the group operations are commutative, the Cayley graph adjacency matrix is symmetric.

$\circ$	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	0	4	5	3
2	2	0	1	5	3	4
3	3	5	4	0	2	1
4	4	3	5	1	0	2
5	5	4	3	2	1	0

(9)

Definition 3.2 A Cayley graph is a graphical representation of an algebraic group. Consider group $(G, *)$ and a subset N of G. The Cayley graph representing G with respect to N is denoted $\mathrm{Cay}(G, N)$. The vertex set of $\mathrm{Cay}(G, N)$ is the elements of G, and the edge contains pairs of elements from G, (x, y) such that $y = x \diamond n$ for some $n \in N$. Note that the group product operation $*$ and the generator operation $\diamond$ are not necessarily the same.

The vertex set of the Cayley graph, as described in Definition 3.2 for a permutation group, consists of vertices corresponding to $s_i \in S$, with edges established by pairs (s_i, s_j). A

generator function determines the assignment of edge weights or colors in the Cayley graph. This function is essential in selecting the specific spectrum to be extracted. The adjacency matrix of the Cayley graph, $A = [e_{ij}]$, is constructed using the generator function outlined in Eq. 10. To apply the generator function, we first perform the symmetry group operation $s_i \circ s_j$ on any two elements $(s_i, s_j) \in S$, followed by the coloring function $f(\cdot)$, which uses the original discrete signal to assign edge weights between vertices representing the elements. If the function outputs zero, no edge is present, resulting in an incomplete graph.

$$e_{i,j} = f\left(s_i \circ s_j\right) \tag{10}$$

In the specific example of $\mathbf{S_3}$, V includes vertices that match the elements described in Eq. 8. These elements representing the vertices are the integers $V = \{0, 1, 2, 3, 4, 5\}$ and the edges are every pair combination of elements $E = \{(s_i, s_j) \mid s_i, s_j \in S\}$. The graph $\text{Cay}(V, E)$ is shaped by the appropriate generator in Eq. 10, which assigns weights to the edges. As a result, the weighted Cayley graph adjacency of the example is shown in Eq. 11, essentially representing connections between all elements shown in the graph and then applying the generator function $f(\cdot)$.

$$A = \begin{bmatrix} f(1) & f(2) & f(3) & f(4) & f(5) \\ f(2) & f(0) & f(4) & f(5) & f(3) \\ f(0) & f(1) & f(5) & f(3) & f(4) \\ f(5) & f(4) & f(0) & f(2) & f(1) \\ f(4) & f(3) & f(5) & f(1) & f(0) \\ f(5) & f(4) & f(3) & f(2) & f(1) \end{bmatrix} \tag{11}$$

The spectrum of a Cayley graph is obtained from the eigenvalues of its adjacency matrix, which are the solutions to the matrix's characteristic equation (Gallian, 2021). For more details on group theory and deeper insights into Cayley graphs, refer to Gallian (2021).

3.2 Related work on Cayley graph spectra

As previously mentioned, the spectral properties extracted by the Cayley graph depend on the group elements and the generator function. Previous work has explained how the Cayley graphs with the symmetry permutation group elements can extract spectra similar to that of the KL spectra for one-dimensional switching functions (Thornton, 2005). Other related works look at Cayley graphs with different generators and group elements to obtain spectra similar to that of other transforms, such as the Walsh spectra (Bernasconi and Codenotti, 1999) and the Chrestenson spectra (Thornton et al., 2002). The main application for analyzing these Cayley graph spectra was to optimize and reduce one-dimensional switching functions. These methods work with multiple valued functions and not just binary-valued switching functions (Thornton, 2005; Thornton et al., 2002). As an example, in previous work (Townsend and Thornton, 2001) on obtaining an equivalent Walsh spectra, the group is described by $(M, \oplus)$, where $M = m_i$ is the set of all possible 2^n minterms for a boolean function f and $\oplus$ is the exclusive-OR operation being applied over a pair (m_i, m_j). Thus, the weighted adjacency matrix of the Cayley graph is formed using the generator $e_{ij} = f(m_i \oplus m_j)$, and the spectrum of the graph is related to that of the Walsh spectrum. Alternatively, as described in Thornton et al. (2002) to obtain the spectra similar to that of the Chrestenson spectra, a different generator $e_{ij} = f(m_i \ominus_p m_j)$ is used, where $\ominus_p$ is the modulo-p operation. The goal of our work

is primarily to extend the previous work defining the relationship of symmetry permutation groups and the 1D KLT (Thornton, 2005) to higher dimensions and provide new applications.

Another approach focuses on applying graph signal processing (GSP) techniques directly to graphs to obtain spectral information. For example, some studies analyze the spectra of directed graphs with orthonormal constraints to construct a digraph Fourier transform (Shafipour et al., 2018). Others convert discrete signals into graphs as a preprocessing step before applying GSP methods (Narang and Ortega, 2011; Sandryhaila and Moura, 2013). More recently, the spectral decomposition of adjacency matrices for weighted Cayley graphs has been studied to design systems such as Frobenius–Schur and Cayley frames, which provide stable and reliable signal reconstruction frameworks (Beck et al., 2024). These methods demonstrate how GSP can be extended to more complex graphs, including Cayley graphs.

A factor influencing the extraction of spectra from Cayley graphs is the process of remapping function indices. Performing remapping of indices of a discrete function, including for higher dimensional images, has a direct impact on the ability of the Cayley graph to maximize decorrelation. This importance of mapping is also discussed in Thornton (2005). Since the Cayley graph involves creating an adjacency matrix from the original data and its permutations, remapping the data to minimize transformations between permutations can thus improve decorrelation and increase the number of zero eigenvalues. Further work on obtaining spectra containing decorrelation information from Cayley graphs in Li and Thornton (2005) examines finding optimal mappings. The research in Li and Thornton (2005) specifically finds that there is a subset of spectra that occur for a certain signal for various mappings. Similarly, an image can mapped to a discrete vector form using many different sets of indices following a permutation operation. Therefore, the optimal higher-dimensional Cayley graph can also be impacted by the permuted mapping of indices. As expected, the remapping of indices is an important component of the maximal decorrelation of a signal and, therefore, is another aspect of Cayley graph generation that can be optimized.

4 Contribution

4.1 Symmetry group representation for N-dimensions

Previous work (Thornton, 2005) focused on extracting the spectra similar to the KL spectra from autocorrelation matrices generated from 1D signals using Cayley graphs. In this section, we show that higher dimensional spectra related to the decorrelation of signals can be extracted from the higher dimensional signal's autocorrelation matrix in Eq. 2 using the Cayley graph created for the $S_n \times S_n$ group. For groups, the direct product operations $\times$ result in a Cartesian product between the different group elements, resulting in pairs of combinations of the two elements. For example, for the direct product of two groups $G_1 \times G_2$, let the elements of group G_1 be denoted by g_1 and g_2, and the elements of group G_2 be denoted by h_1 and h_2. The elements of the direct product group $G_1 \times G_2$ consist of ordered pairs of elements from G_1 and G_2, such as (g_1, h_1) and (g_2, h_2). The group product operation in $G_1 \times G_2$ is defined component-wise. That is, for any two elements (g_1, h_1) and (g_2, h_2) in $G_1 \times G_2$, their group product is given by $(g_1, h_1) * (g_2, h_2) = (g_1 * g_2, h_1 * h_2)$, where $g_1 * g_2$ represents the group product operation in G_1, and $h_1 * h_2$ represents the group product operation in G_2. This group notation can represent multiple independent operations across multiple dimensions with a single group operation.

We now show that by using direct products of groups, the new combined group product operator can be the permutation operator $\circ$ and now represent permutation for multiple axes. We can use group elements from the $S_n \times S_n$ group to represent the same shifting operations we see in the columns of the 2D autocorrelation matrix. In this case, the elements $s_i \in (S_n \times S_n)$ can be used to represent valid 2D shifting permutations as numerical values 1, 2, .. and are defined by operations representing unique but valid image shifts across multiple dimensions. By definition, the direct product group now consists of pairs of elements. Let us represent the group of row permutations, where $r_i \in S_n$ permutes the rows, and the group of column permutations, where $c_i \in S_n$ permutes the columns. The elements of the combined group are pairs $s_i = (r_j, c_k)$, where j and k could represent independent shifts along the corresponding x and y axes. This subset of permutations can still be represented with the $\circ$ operator since it shifts a subset of the symmetry permutation group. Note that unlike the previous symmetry group Table 9, the combination of permutations using the operation $\circ$ is commutative since an image can be shifted along both x and y axes and the shifts accumulate the same way regardless of order. For $s_i \in S_n \times S_n$, $s_i = (r_j, c_k)$, group operation $\circ$ is applied component-wise. For example, for two elements $s_1 = (r_i, c_i)$ and $s_2 = (r_j, c_j)$, the product is given by: $s_1 \circ s_2 = (r_i \circ r_j, c_i \circ c_j)$. Here, $r_i \circ r_j$ is the group product of the row permutations, and $c_i \circ c_j$ is the group product of the column permutations. These s_i are the group elements representing the different valid permutation actions on the image. In the standard permutation notation $s_i \circ s_j = s_k$, we can show that multidimensional permutation s_i then permuted by s_j can be simplified as a single permutation s_k.

We give an example for the 3×3 image or $S_3 \times S_3$ Cayley table in Eq. 12. The elements can be represented in terms of the cyclic permutations $s_0 = (r_0, c_0)$, $s_1 = (r_1, c_0)$, $s_2 = (r_1, c_1)$, $s_3 = (r_2, c_1)$.... Following this logic, we can understand that combining different permutations, such as s_0 and s_0, will result in $(r_0, c_0) = s_0$ since no rows or columns have been shifted. For a non-trivial example, we can see that $s_8 \circ s_8 = (r_2, c_2) \circ (r_2, c_2) = (r_{(2+2) \mod 3}, c_{(2+2) \mod 3}) = (r_1, c_1) = s_4$. In simple terms, making the same s_8 permutation twice results in an equivalent s_4 operation; this can be verified by using Eq. 12.

$\circ$	0	1	2	3	4	5	6	7	8
0	0	1	2	3	4	5	6	7	8
1	1	2	0	4	5	3	7	8	6
2	2	0	1	5	3	4	8	6	7
3	3	4	5	6	7	8	0	1	2
4	4	5	3	7	8	6	1	2	0
5	5	3	4	8	6	7	2	0	1
6	6	7	8	0	1	2	3	4	5
7	7	8	6	1	2	0	4	5	3
8	8	6	7	2	0	1	5	3	4

(12)

It is important to note that this group consists of symmetry group elements representing cyclic shifts, which results in a symmetric adjacency matrix. Note that this symmetry permutation group in 2D can readily be extended to elements representing higher dimensional permutations, known as the k-fold product of S_n, denoted as $(S_n)^k = S_n \times S_n \times ...S_n$ (k times). Unlike the standard autocorrelation matrix, this adjacency matrix will be symmetric even when representing higher dimensional permutations. This group representation extends the symmetry permutation group in the previous work (Thornton, 2005) to higher dimensions through the direct products of groups.

The adjacency graph can be described by the formula $e_{ij} = f(s_i \circ s_j)$, where the edge weights of the adjacency graph are defined by the group operation $\circ$ on every combination of group elements s_i, s_j and the application of coloring function $f(\cdot)$. The generation of the Cayley graph adjacency matrix can be described by referencing the operator combinations of $s_i \circ s_j$ in the Cayley table in 12, which has the values $0-8$, and using those values to index the flattened image function $\mathbf{I_f}(\cdot)$. To create the image function, we flatten the original image by taking its pixel rows and concatenating them side by side. This function application is similar to the example in Eq. 11 that applies the generator function in Eq. 10 to the previous table in Eq. 12.

4.2 Cayley graph generation

Instead of using a Cayley graph to represent combinations of simple image shifts, we can use a Cayley graph to represent the relationship among all the permutations found in the columns of an autocorrelation matrix on a set of image indices. To do this, we use $\mathbf{F}$ autocorrelation in Eq. 2 from Sect. 2.1 and replace $\ominus$ with the symmetric permutation group operator $\circ$ discussed in Sect. 4.2. For each column, the multidimensional indices $(x_0, x_1, ...x_N)$ of an N-dimensional signal are permuted by one of the corresponding $s_0, s_1 ... s_n$ elements of the symmetry permutation group $(S_n)^N$. To simplify the notation, we can present the unique high dimensional index elements as i and the unique permutation elements on the multiple dimensional axes as j. Thus, the action of the permutation on the indices is $i \circ j$, representing the multidimensional permutations of the high dimensional indices and, therefore, the shuffling of the flattened multidimensional image function $\mathbf{I_f}(\cdot)$ as each column.

$$\mathbf{F}_\circ = [\mathbf{I_f}(\mathbf{i} \circ \mathbf{j_0}) \quad \mathbf{I_f}(\mathbf{i} \circ \mathbf{j_1}) \ldots \mathbf{I_f}(\mathbf{i} \circ \mathbf{j_{N-1}})] \tag{13}$$

We have demonstrated that the $\mathbf{F}$ autocorrelation matrix can be expressed using higher-dimensional symmetric group operations, implemented as shifting permutations on the original image indices, allowing it to be modeled as a Cayley graph. The generator function from Eq. 10 is employed to construct this Cayley graph by applying the symmetry permutation operator $\circ$ between each image index and a valid permutation group element. In this Cayley graph structure, i is treated as the starting node and j as the ending node for a directed edge. In the pair (i, j), i corresponds to the image index element s_i, while j represents a node associated with a valid permutation element s_j. The edges that link image indices to different permutations follow the notation $i \circ j$, reflecting how the original image index i is permuted based on the permutation element j. The graph is created by connecting every node representing an image index to every node representing a permutation element, forming edges (i, j), with edge colors determined by the generator function from Eq. 10. The function $f(\cdot)$ represents the original flattened multidimensional signal $\mathbf{I}_f(\cdot)$ and is applied to the edges.

We can also look at the creation of the graph adjacency programmatically at the pixel level through the indexable form of the generator $e_{i,j} = \mathbf{I_f}^j(i)$. To help with programming the pixel-level definition, we first look at the $\mathbf{F}$ pixel-level autocorrelation definition in Eq. 14 but with the i values indexing the specific pixels of the selected permuted function $\mathbf{I_f}^j$. Thus, this function can be used to access the precomputed i-th row and j-th column of the autocorrelation matrix. Using the matrix definition that shows each permutation as j and the specific pixel at the location as i. This gives us the function definition $\mathbf{I_f}^j(i)$ that simplifies the creation of the Cayley graph. From the precomputed autocorrelation, the corresponding

weights for directed edge nodes (i, j) are passed to the unpermuted coloring function $e_{ij} = \mathbf{I_f}^j(i)$. As discussed in Sect. 4.2, edges with either the original or permuted pixels being zero would result in the edge not being present. We can keep nonzero edges as described by $E = \{i, j | \mathbf{I_f}^j(i) > 0\}, i, j \in V$, where the permuted image function $\mathbf{I_f}^j(i)$ indexed by i are used to remove edges with zero values.

$$\mathbf{F} = \begin{bmatrix} \mathbf{I}_f^{j_0}(i_0) & \cdots & \mathbf{I}_f^{j_{m-1}}(i_0) \\ \vdots & & \\ \mathbf{I}_f^{j_0}(i_{m-1}) & \cdots & \mathbf{I}_f^{j_{m-1}}(i_{n-1}) \end{bmatrix} \tag{14}$$

The creation of the Cayley graph is described in Algorithm 1. The Algorithm 1 clearly shows that the adjacency graph can be created from the 2D autocorrelation to assign edge weights between every two group elements s_i and s_j. The algorithm uses $\mathbf{F}$ from Eq. 14, which is the precomputed N-D autocorrelation. The color generator f is the j selected column of F[:,j] and is used to color the edge from that s_j permutation node to the s_i index node. The directed Cayley graph adjacency can be generated using three types of lists: source nodes, which represent image index elements; target nodes, which represent permutation elements; and edge colors, derived from the permuted function.

Algorithm 1 Generating Cayley graph of symmetric group elements

```
for j = 1 to Nrows do
    f = F[:, j]                          ▷ Select permuted function
    for i = 1 to length(f) do
        if f[i] > 0 then
            s_i ← append(s_i, i)         ▷ Index node
            s_j ← append(s_j, j)         ▷ Permutation node
            e ← append(e, f[i])          ▷ Colored using function
        end if
    end for
end for
```

In our definition, we noted there will be some vertices without colored edges, meaning a node-to-node edge e_{ij} will sometimes be zero. Removing nodes from the adjacency graph that have no edges can reduce the size of the adjacency matrix and represent the compression of the data without the loss of any information. The dropping of nodes without connections, with a degree of zero, can be described here: $V' = V \setminus \{v \in V : d(v) = 0\}$. This can also be quickly analyzed in constant time by seeing where occurs when both the $e_{x,y} = e_{y,x} = 0$. These nodes have no information between the original pixels and the permutations since there are no colored edges. We have shown that obtaining eigenvalues of the reduced adjacency matrix after dropping the nodes without connections only removes the zero eigenvalues and does not impact the magnitudes of the other eigenvalues, making the dropping of these nodes analogous to the reduction of basis vectors with corresponding eigenvalues of zero in the KL spectra. We can further extend pruning to removing lower non-zero degree nodes as well; as we will see in the next Sect. 5, removing the lower degree nodes results in removing lower magnitude eigenvalues. Removing lower-degree nodes will remove node elements representing index nodes with very few edges, corresponding to signal components that are infrequently accessed by permutations and, therefore, have less information. We leverage

this to prune lower-degree nodes before eigendecomposition, eliminate lower-magnitude eigenvalues, and reduce matrix sizes for the autocorrelation representation.

By using a Cayley graph representation, only the permutation nodes, index nodes, and the weights between them colored by the image are required. This weight coloring method colors the paths from the node representing each image pixel index to another node representing each unique permutation, with a weight value selected from the permuted coloring function. This method represents the autocorrelation matrix information and its permutations in an alternative graph form by representing each pixel and its permutations as group elements of a Cayley graph adjacency. As we will see in the next section, the properties of the Cayley adjacency allow us to extract the spectra capturing decorrelation properties.

5 Experimental evaluation

5.1 Spectral analysis

We now analyze the ability of the CG to efficiently extract spectra containing decorrelation information. One of the differences between our method and the KL transform is that rather than representing variance information through the multiplication of signal components, the CG instead represents the relationship between different indices and their permutations. From the CG spectra, we can analyze the degree to which the set of permutations varies the signal's components. For example, if a specific index node is connected to multiple permutation nodes with edges of the same value, this indicates that permutations do not impact the variation at that component and, as a result, indicate low variation. Since the connections from the index node to its permutations are all similar in value, the clustering of similar nodes would result in zero values in the spectra of the adjacency matrix, a property commonly seen in spectral clustering (Ng et al., 2001). In spectral clustering, the spectra contain a higher number of zeros when subgraphs have distinguishing patterns that can be clustered and represented by a few eigenvectors. To apply spectral clustering to a graph, the Laplacian matrix is typically computed to ensure the matrix is positive semidefinite. However, in our work, we simply find the absolute magnitudes of our negative eigenvalues, as our method achieves sparse spectra without the additional computation of the Laplacian. As discussed in Sect. 3.2, the remapping of function indices has an impact on the decorrelation of the signal when using Cayley graphs. By grouping similar components together through remapping, we can minimize the impact of permutations, and the signal can be more easily decorrelated, resulting in a sparse spectrum similar to what is observed in spectral clustering.

To test our method, we create a 3×3 image containing random floating-point values between zero and one, and we generate the corresponding CG of size 49×49. This larger size results from representing the various permutations of each signal component, including padding to account for all overlapping shifts, ensuring a minimum overlap across boundaries. For example, for the 3×3 image, padding is applied to achieve at least a 1-pixel overlap on each side. This padding expands the image dimensions to 7×7, as the original 3×3 image requires this larger grid to accommodate overlaps on all sides. Consequently, the padded structure increases the effective graph size needed to represent all possible permutations. Using this image, we demonstrate how the ability of our method to decorrelate a signal is influenced by different index mappings, as demonstrated in Fig. 2, which analyzes the spectra of a test image under various mappings. These spectra, derived from only 16 pixels and their permutations, create a dense representation of information. As previously mentioned,

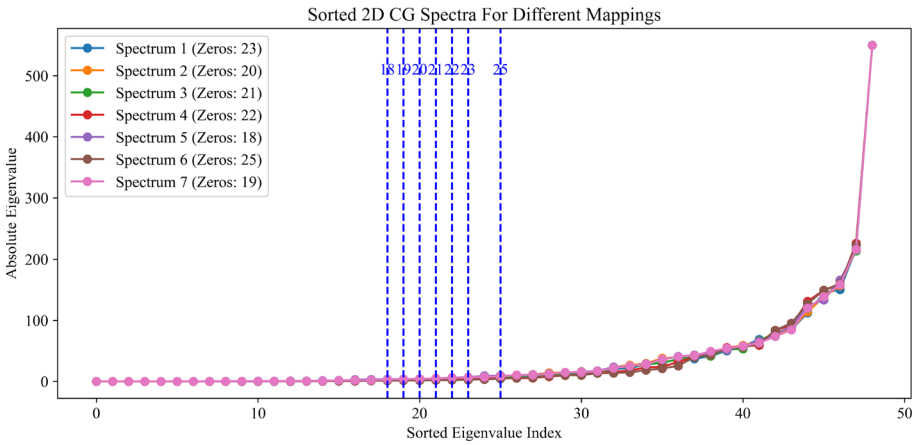


Fig. 2 Sorted 2D CG spectra for various index mappings of a 3×3 image, showing absolute eigenvalues on the y-axis and their sorted index on the x-axis. Vertical blue dashed lines indicate the number of zero eigenvalues for each mapping

remapping the function indices results in different numbers of zeros, indicated by the vertical blue dotted lines. Although the number of zeros varies, the structure of the few largest eigenvalues remains consistent. This consistency enables us to use these eigenvalues as robust features for various applications, similar to how the largest values in the KL spectra capture the majority of the representation information. Therefore, while index mappings affect the number of zeros, the largest eigenvalues, similar to those in KL spectra, retain most of the signal's information.

Figure 3 shows a 3×3 test image with intensities $1 - 9$. The corresponding Cayley graph adjacency matrix (CG) for the 3×3 example is shown in Fig. 4, where every pixel intensity represents an edge coloring. The Cayley graph representation extracts the decorrelation information using the permutations of the original image using the method described in Sect. 4.2, where each element represents a permutation from the $S_n \times S_n$ group. For visualization purposes, Fig. 5 is the CG adjacency matrix after pruning the 20 least connected nodes. The methods for generating this adjacency matrix and pruning nodes are described in Sect. 4.2, and we will show in the next section the benefits of pruning.

We can extend our CG method to N dimensions as shown in Sect. 4.2. As an example, we first represent the 3D Cayley graph adjacency matrix of a 3D volume and constructed through group operators connecting nodes that represent elements in the $S_n \times S_n \times S_n$ group. For our first 3D test image, we create the $3 \times 3 \times 3$ block with intensities ranging from $1 - 27$. The 3D image is shown in Fig. 6 divided into slices along the Z-axes, so every 2D slice is shown as an image. The corresponding CG representing the 3D volume is shown in Fig. 7. As expected, the Cayley graph size increases with the number of axes as a result of more permutations caused by shifting along multiple axes. Figure 8 more easily visualizes the structure of the same CG after removing the 200 least connected nodes.

For visualizing the impact of pruning on the CG spectra, a random 15×15 matrix is used, where each pixel is assigned one of five possible integers (1 through 5) with equal probability. As shown in Fig. 9, the eigenvalues of the Cayley graph's adjacency matrix (size 1850) decrease exponentially in magnitude. This suggests that most of the variance is captured by the initial few eigenvectors. Given that the Cayley adjacency matrix has

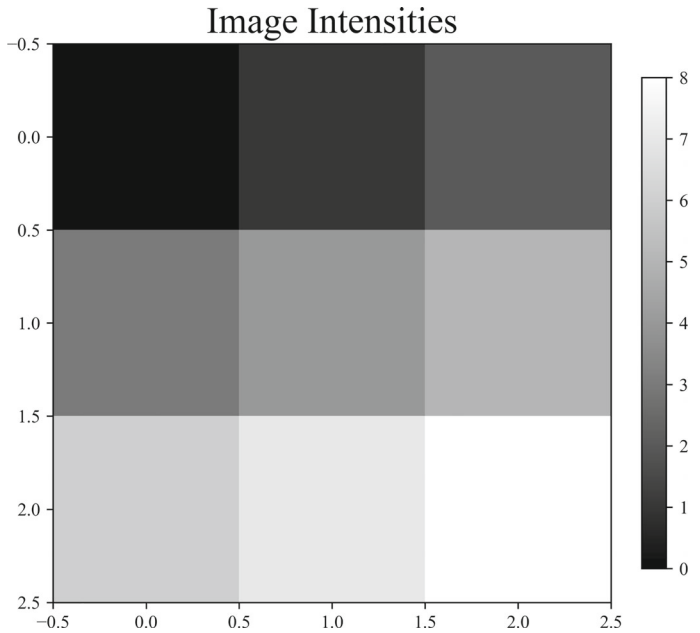


Fig. 3 Small 3 by 3 image for testing

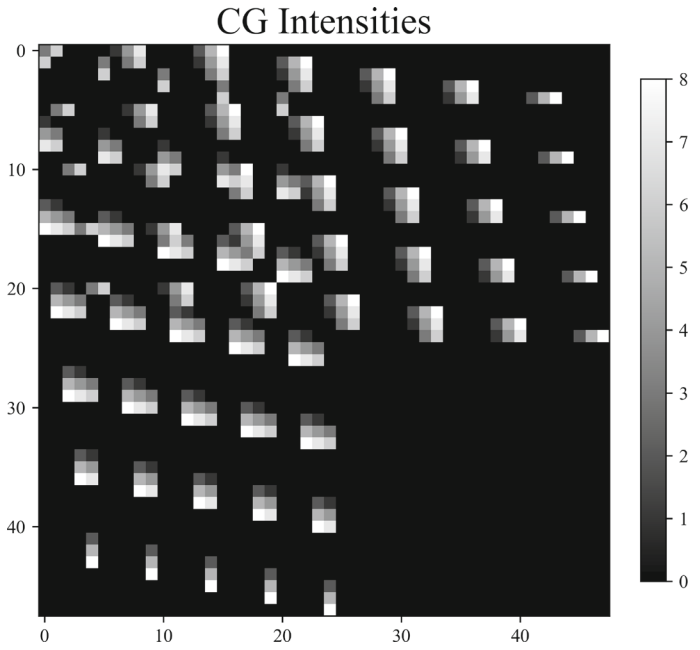


Fig. 4 Cayley adjacency for the 3x3 image

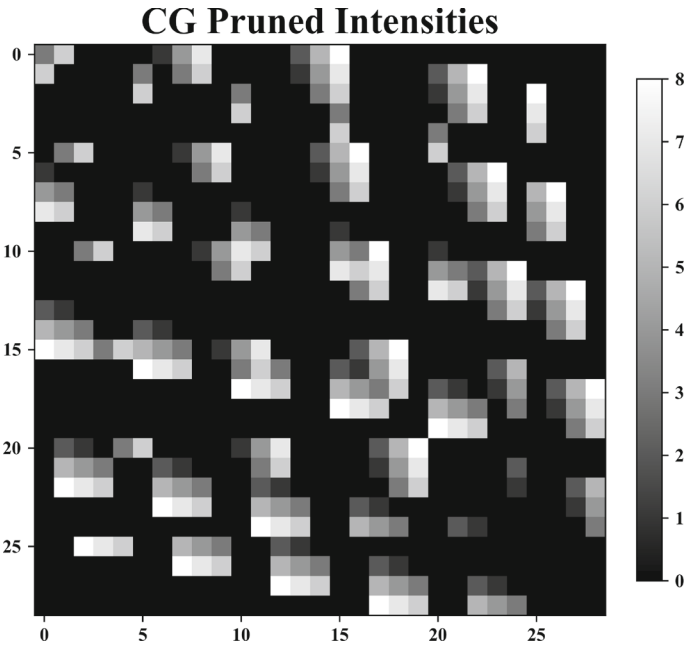


Fig. 5 Pruned Cayley adjacency

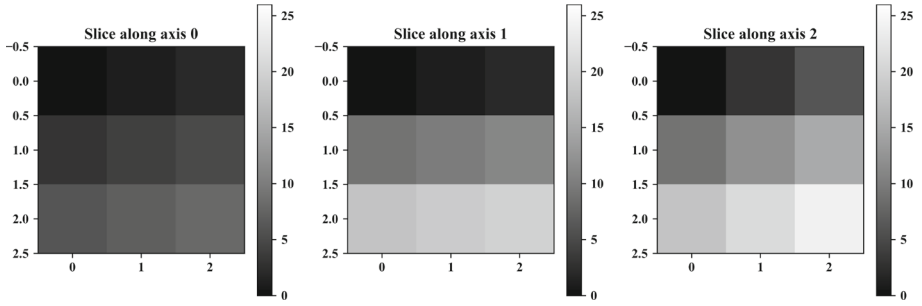


Fig. 6 3D test image as three 2D slices

orthogonal eigenvectors and exponentially decreasing, sparse eigenvalues, the spectrum of the Cayley graph behaves similarly to that of the KL transform.

As illustrated in Fig. 9, pruning the least connected nodes does not affect the highest eigenvalues–those tied to the eigenvectors capturing the majority of variance–until roughly 650 of the original 1850 nodes are pruned. This suggests a strong link between the lower-degree nodes and the lower-magnitude eigenvalues and eigenvectors of the Cayley graph. This behavior indicates optimized decorrelation since the larger eigenvalues correspond to features with significant information. Our method takes advantage of this by pruning lower-degree nodes before performing eigendecomposition, which helps remove lower-magnitude eigenvalues and reduces the memory required to represent the adjacency matrix.

In Fig. 9, we visually show that the graph pruning process only reduces the number of zero and low magnitude eigenvalues. The figure shows some decorrelation properties since we maintain most of the spectral information while reducing the graph size by removing the

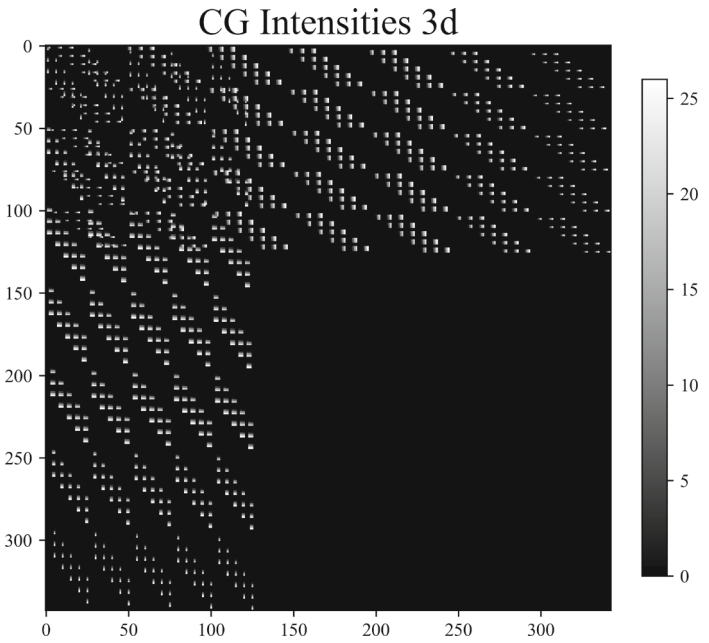


Fig. 7 3D Cayley graph adjacency

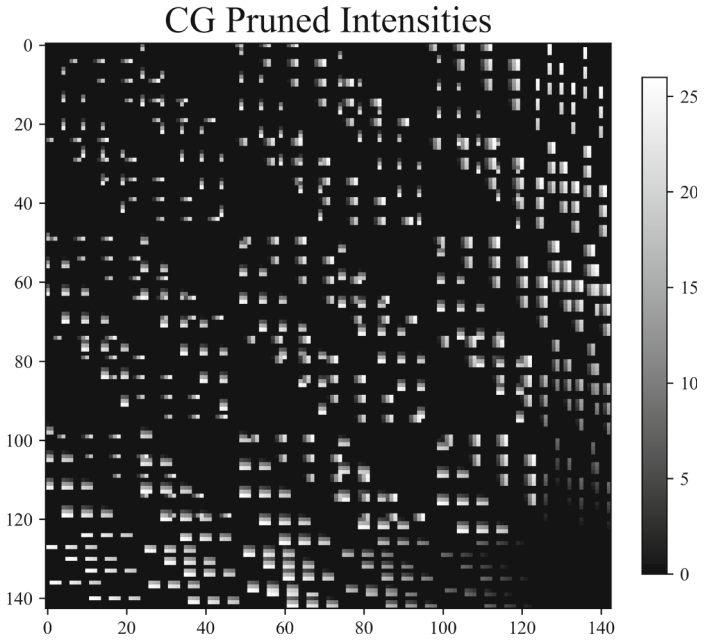


Fig. 8 Pruned 3D Cayley graph adjacency

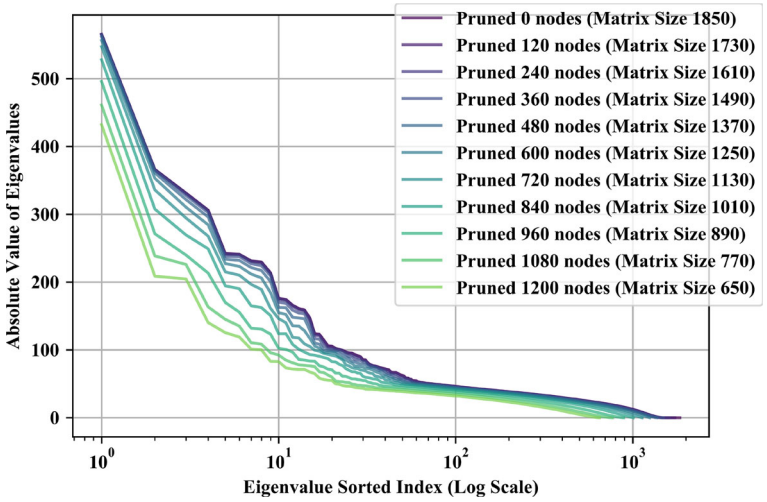


Fig. 9 Log-scaled eigenvalue index for CG spectra across different pruning levels, showing the effect of pruning on reducing low-magnitude and zero eigenvalues. The spectral information of higher-magnitude eigenvalues is preserved while the graph size decreases

nodes that impact the zero eigenvalues. As an application of the pruned CG, in Sect. 5.2, we show we may obtain equivalent higher-magnitude eigenvalues with fewer calculations and a smaller memory footprint.

5.2 Time and space analysis

In higher dimensions, the adjacency size of the Cayley graph grows exponentially, scaling as n^d for d dimensions with n possible permutations per dimension. This exponential growth in adjacency size arises because the Cayley graph connects each component value to all others through valid permutations. Although the Cayley graph adjacency matrix is larger than an autocovariance matrix, our method reduces the number of required multiplication operations. In contrast, in the KL transform autocovariance calculation, the number of multiplications grows exponentially at n^d. In this case, n is the number of signal components, where each n-th component in every dimension must interact with all others, as outlined in the autocovariance definition in Eq. 5. This growth reflects the requirement that each signal component interact with every other component through multiplication. By avoiding these multiplications, our method generates the CG spectra more efficiently than the KL spectra despite requiring additional space requirements. In our experiments, we focus on efficiently obtaining the first few eigenvalues and minimizing the impact of large graph sizes through pruning.

We evaluate performance by measuring both the matrix generation time and the time required to compute the two largest eigenvalues by magnitude. The eigenvalues are computed using the method outlined in Lehoucq et al. (1998). Additionally, parallelizable divide-and-conquer algorithms for eigenvalue computation of symmetric matrices, described in Bientinesi et al. (2005), offer potential improvements. It is important to note that the timing results for the eigendecomposition are directly proportional to the size of the matrices.

To test performance, we use randomized example images of six sizes, 5 to 30 in increments of 5, to generate Cayley graph adjacency matrices and autocovariance matrices. The results,

Table 1 Performance metrics for autocovariance matrices of randomized images with sizes ranging from 5 to 30 (increments of 5)

Image size	Eigs (ms)	Generation (ms)	Total (ms)	Matrix size
5	0.98	4.08	5.06	25000
10	1.05	59.69	60.74	100000
15	1.64	358.41	360.06	225000
20	1.63	1135.40	1137.03	400000
25	2.24	2871.74	2873.98	625000
30	2.68	5583.56	5586.24	900000

The table includes eigenvalue computation time (Eigs), matrix generation time, total time, and matrix size. Results are averaged over 30 randomized test images per size

Table 2 Performance metrics for Cayley graph adjacency matrices of randomized images with sizes ranging from 5 to 30 (increments of 5)

Image size	Eigs (ms)	Generation (ms)	Total (ms)	Matrix size
5	2.81	0.48	3.29	170000
10	4.43	7.06	11.50	785000
15	64.02	45.23	109.25	1850000
20	272.63	132.75	405.38	3365000
25	739.29	304.20	1043.49	5330000
30	1610.73	654.95	2265.68	7745000

The table includes eigenvalue computation time (Eigs), matrix generation time, total time, and matrix size. Results are averaged over 30 randomized test images per size

summarized in Table 2 for the Cayley graphs and Table 1 for the autocovariance matrices, are averaged over 30 random images for each size, with each pixel randomly assigned one of five possible integer values $(1-5)$ with equal probability. For consistency, we used the same test set of matrices for both autocovariance and CG comparisons to ensure a fair timing evaluation.

The data shows that autocovariance matrices are considerably smaller compared to CG adjacency matrices, with autocovariance matrix size growing quadratically relative to image size. The CG adjacency matrix size also grows at a quadratic rate with respect to image size but with a larger scaling coefficient. Consequently, the time required to compute the two largest eigenvalues (shown under the 'Eigs' column) is significantly shorter for autocovariance matrices. However, as expected, the generation time for autocovariance matrices is longer due to the increased number of multiplications required in higher-dimensional signals. This additional generation time of the autocovariance matrix offsets the computational advantage of faster eigenvalue calculations for smaller matrices. We note that for larger matrices, further testing is needed to obtain the time required to compute the eigenvalues.

We visualize the data from the previous tables in a series of plots to directly compare the time and space complexity for each method. Figure 10 shows the time required for matrix generation and eigenvalue computation, while Fig. 11 illustrates the growth of matrix sizes. When comparing the time required for eigendecomposition, the CG method exhibits a higher curve, indicating worse performance than the autocovariance method. However, when considering the total time, which includes matrix generation and eigendecomposition, the CG method

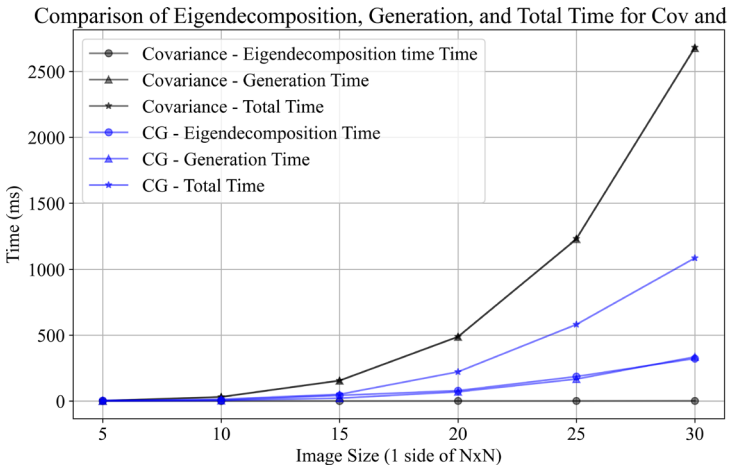


Fig. 10 Comparison of matrix generation, eigendecomposition, and total computation times for covariance (Cov) and Cayley graph (CG) methods as a function of image size. While the CG method has higher eigendecomposition times due to larger matrices, its faster matrix generation leads to a lower total computation time compared to the covariance method

demonstrates a lower overall curve, reflecting faster performance. The longer eigendecomposition times are a direct result of the larger CG matrices. As anticipated from Sect. 4.1, CG matrices demonstrate a polynomial growth rate in matrix size. For example, when the image size doubles from 10×10 to 20×20, the autocovariance matrix size increases from $100,000 \times 100,000$ to $400,000 \times 400,000$, while the CG matrix size expands from $785,000 \times 785,000$ to $3,365,000 \times 3,365,000$. This indicates that the CG and autocovariance matrices increase fourfold when the image size doubles, although the CG method begins from a higher initial value. To mitigate this growth rate, the applications described in Sect. 6 apply the CG method over smaller, constant-sized windows, reducing both matrix size and computation time. Overall, we see that when we use our CG method with these smaller windows, the CG method is faster due to the lower matrix generation time.

In Table 3, we show that pruning reduces the size of the CG adjacency matrix, leading to faster eigenvalue computation. We run the CG and pruned CG methods on the same image set, with the settings described earlier being 30 images per size, random values 1-5. The reduction in matrix size by pruning can be shown by the "Matrix Size Prune" column having lower values than the Table 2, being 20% smaller. We show in column "Eigs % diff" that there is around a less 3% difference between the eigenvalue ratio of the two largest eigenvalues between the pruned and original CG spectra. Overall, these results highlight that additional post-processing, such as pruning, effectively reduces the impact of eigendecomposition on total time, particularly when only the largest magnitude eigenvalues are needed. This approach can help optimize the CG method's performance.

6 CG spectra applications

In this section, we show that possible applications of using our CG spectra include spectrogram classification, 2D feature matching, and 3D boundary detection. The CG method's performance is compared to that of well-established analytic methods. While many high-

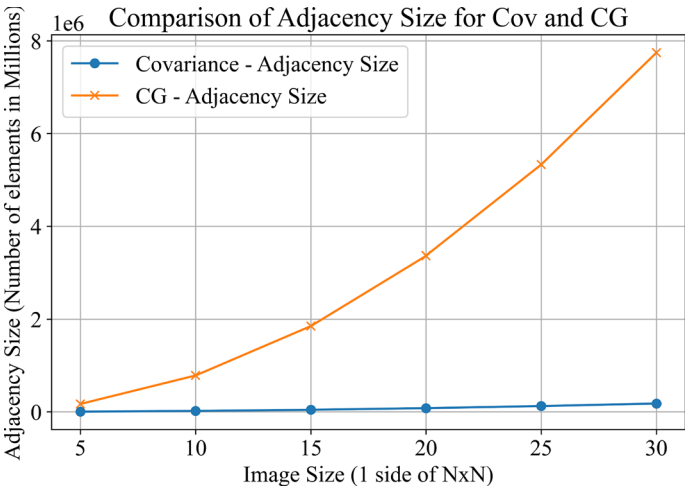


Fig. 11 Growth of adjacency matrix sizes for covariance (Cov) and Cayley graph (CG) methods as the image size increases. Both methods demonstrate quadratic growth in matrix size, with the CG matrices starting from a higher initial value and scaling with a larger coefficient compared to covariance matrices

Table 3 Comparison of Cayley graph (CG) and pruned CG methods for adjacency matrices of varying image sizes

Image size	Eigs time (ms)	Eigs time prune (ms)	Matrix size	Matrix size prune	Eigs % diff
5	5.44	3.90	170000	136000	7.95
10	8.39	6.36	785000	628000	3.23
15	64.52	41.35	1850000	1480000	2.85
20	287.76	197.39	3365000	2692000	2.73
25	743.95	542.53	5330000	4264000	2.36
30	1574.29	1098.50	7745000	6196000	2.11

The table includes eigenvalue computation time (Eigs Time) and its reduction after pruning (Eigs Time Prune), matrix size before and after pruning (Matrix Size and Matrix Size Prune), and the percentage difference in the ratio of the two largest eigenvalues between the pruned and original CG spectra (Eigs % Diff)

performance signal processing and edge detection techniques exist, our intent here is not to compete with these methods. Instead, we provide accessible examples to demonstrate the Cayley graph spectra are related to the KL spectra. Applications are not limited to those in this section, and many other applications that use decorrelated eigenfeatures may also be possible, including corner detection, protrusion detection, template matching, and cross-correlation (Brunelli, 2009; Harris and Stephens, 1988; Kok and Rajendran, 2018).

6.1 Signal classification

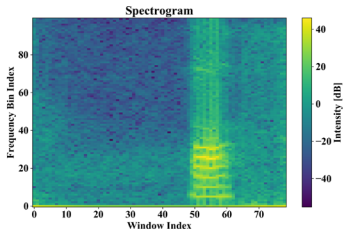
In Sect. 2.2, we noted that the KLT optimally decorrelates the signal into basis functions, and in Sect. 4.2, we emphasized that CG spectra can be more efficiently calculated than the KL spectra for multidimensional signals. This is notable since, as described previously, obtaining the DFT components is more efficient, but the eigenvectors are not optimized for decorrelating the signal. Therefore, the improvement in algorithmic efficiency described

in Sect. 5.2 combined with better decorrelation further increases the ability to use the CG spectra over the other spectra. We take advantage of spectral properties obtained from the Cayley graph and use them as valuable information for signal classification. Frequency-based methods are limited by the assumption that critical features exist in distinct frequency bands separate from noise, an assumption that fails when noise and signal overlap in the frequency spectrum. The Cayley graph does not rely on frequency-based information and instead relies on the spectral coefficients being directly related to the decorrelation properties; in this case, the largest N eigenvalues of the autocovariance will represent the amount of variation captured by the first N principal components. The smaller variance in the noise will not greatly impact these components since most of the magnitude of the components comes from the larger, meaningful variance in the data itself. As a result, we expect the CG transform coefficients to contain valuable classification features that are noise-resilient and do not rely on distinguishing frequency information. Another property that we will notice is that since the CG transform maximizes zero value coefficients, only the N nonzero coefficients are needed for classification. In contrast, for the frequency-based methods, we need to choose a set of frequency bins to work with based on the application. The result is that our transformation can be greatly reduced to only storing the N largest components, though, for our work, we maintain an equal comparison by making sure the number of frequency bins aligns with the number of components of the CG transformed signal.

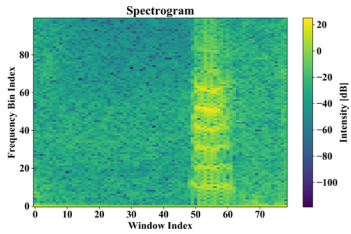
To start the classification task, we apply the CG transform to generate spectrograms from one-dimensional signals by using it on two distinct types of 1D data: a subset of audio signals from the Google Speech Commands dataset (Warden, 2018) and ECG signals from the MIT-BIH Arrhythmia dataset (Moody and Mark, 2001). We demonstrate the ability of the CG approach to extract features by evaluating the classification accuracy of our method and comparing it against other commonly used transform-based methods. Spectrograms are generated using three different transform techniques: the discrete cosine transform (DCT), the fast Fourier transform (FFT), and our proposed CG transform. For the audio task, we also apply a mel-scaled spectrogram, a well-known technique in audio classification. For both the audio and ECG tasks, signals were downsampled to final rates of 8,000 Hz and 2,000 Hz, respectively. Each signal was padded to match the longest sample in the dataset to ensure uniform input data for the machine learning models. The spectrograms were created with a window size of 200 samples and a 50% overlap to capture detailed frequency information while maintaining consistency between successive windows. For the audio task, the mel-scaled spectrogram was constructed using 100 mel frequency bands, ensuring a fair comparison with the 100 frequency bins from the DFT. Similarly, for the CG transform, the 100 largest magnitude eigenvalues were selected from the Cayley graph representation of the signal. The spectrogram dataset was split randomly into 70% training and 30% testing sets to ensure sufficient data for robust training and evaluation of the models.

The spectrograms for the various methods applied to the first sample of the Google Speech Commands dataset for the word "no" are shown in the following figures: Fig. 12a presents the DFT spectrogram, Fig. 12b shows the DCT spectrogram, Fig. 12c shows the mel-scaled spectrogram, and Fig. 12d displays the CG transform spectrogram. Within the frequency-based spectrograms of the DCT and DFT, we note that there is a large amount of random variation in frequency at any time window; in comparison, the decorrelation information of the Cayley graph has many zero values since the concentration of the magnitudes is in fewer components.

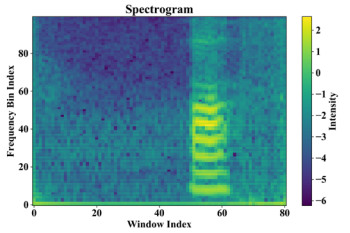
The resulting spectrograms are then flattened and classified using a support vector machine (SVM) model with a linear kernel. We conducted an audio classification task focused on a binary classification problem: distinguishing between the spoken words "on" and "off" with



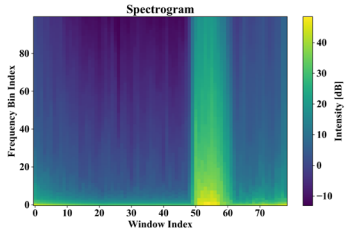
(a) DFT spectrogram for sound sample



(b) DCT spectrogram for sound sample



(c) Mel-Scaled spectrogram for sound sample



(d) Cayley graph spectrogram for sound sample

Fig. 12 Comparison of different spectrograms for the sound sample: **a** DFT spectrogram, **b** DCT spectrogram, **c** Mel-Scaled spectrogram, and **d** Cayley graph spectrogram

1000 samples for each class. The CG transform spectra performed very similarly to the mel-scaled spectra at around 80% classification accuracy. The DCT and FFT methods were similar at only around 68%. We applied the same methodology to the MIT-BIH Arrhythmia dataset, which contains between 200 and 1000 ECG signals for each of the five different classes. The CG transform achieved a classification accuracy of 96.59%, similar to 96.33% for the DCT and 95.77% for the FFT. Overall, the CG transform offers an alternative approach, matching the performance of mel-scaled spectra in audio classification tasks and outperforming traditional methods such as DCT and FFT in both audio and ECG signal classification, all while requiring minimal preprocessing.

6.2 Image feature matching

Feature matching is a prerequisite in computer vision for tasks including object detection, image alignment, and 3D reconstruction. For the task of feature extraction, the decorrelation information of the spectra is useful since the spectral coefficients are invariant to rotation and translation, making them ideal for generating feature vectors that can be robustly used in feature-matching tasks. The invariance of the spectra occurs because the principal components, axes of maximal variation within the dataset, are rotated with the same matrix as the data during an image transformation. As a result, the coefficients that represent the amount of variation captured do not change since the decorrelation and variance information are the same. Since the features of the CG spectra are invariant to translation and rotation, they can be used in feature-matching scenarios where the image is translated or rotated. In addition, the advantage of the CG spectra is that fewer coefficients are needed since the signal energy is concentrated in the fewest possible non-zero coefficients. This allows us to select the largest

N coefficients as our feature vector with minimal loss of decorrelation information. Minimizing feature vectors allows for faster feature matching by reducing the size of the feature vector calculation to find the similarity between vectors. This is especially useful as we can readily extend our detection methods to higher dimensions without necessarily increasing the feature vectors at the same rate. This invariance, combined with the fact that only the largest spectral coefficients are needed, means we can create small but valuable feature vectors that show performance gain over standard methods in our results.

We can use the eigenvalues of our Cayley graph as features for every 3×3 sliding window. These unique eigenvalue responses can be used as feature descriptors for every pixel at the center of the windows. These feature descriptors can then be used to match pixels after transforming the image. Since the feature vectors are computed for every pixel, this task is referred to as dense feature matching. Dense feature matching involves finding correspondences for many pixels or regions across different images. Unlike sparse feature matching, which focuses on a few key points (e.g., corners, edges), dense matching aims to match every pixel or a dense grid of points across images. This dense matching approach is commonly used in tasks that require more detailed and seamless alignment of images, such as optical flow, stereo vision, and image registration. Note that for other applications, such as computing the transformation matrix, only four non-collinear point correspondences are required.

The dense feature matching workflow involves computing the feature descriptors for every pixel, using an initial brute force matching based on Euclidean distance, and finding the exact matches. We must compare dense feature extractors such as *DAISY* (Dense Adaptive Image Similarity) and *Dense-SIFT* (Scale-Invariant Feature Transform). This way, every pixel in the image has a feature descriptor. To compare our method for feature matching, we can simply compare how many correct matches exist before and after the rotation and translation of the image.

Standard SIFT does not extract features at every point; SIFT normally only extracts features at key points defined by the high response areas found by repeatedly blurring and subtracting the image (Juan and Gwun, 2009). The SIFT descriptors at each key point are created from the image gradients. The SIFT descriptors are made rotation and scale invariant through first preprocessing each location with different normalization and orientation methods before obtaining the feature vector. The feature vector represents a histogram of image gradients, with each vector element being a bin for a specific different gradient direction. For our experiments, we use a modified version that finds SIFT descriptors at every pixel instead of only key points. Using SIFT for comparison is valuable as it provides insight into the baseline level of matches achievable with the most popular feature extractor.

A more direct comparison for dense matching can be made to DAISY, which is specifically designed as a dense feature descriptor and has a vector at each pixel (Tola et al., 2009). For this reason, we expect the DAISY feature extractor to perform better than simply applying the dense version of SIFT. DAISY uses image gradients similar to SIFT but also uses Gaussian filter responses instead of histograms of image gradients. Feature descriptors consisting of image gradients for DAISY for each pixel are obtained over circular windows, distinguishing them from the box-shaped windows used by SIFT. The feature descriptor vector for DAISY is also smaller than that of SIFT, containing only 104 elements instead of 128.

Since the CG is generated using symmetric group operations, the eigenvalues result in permutation-invariant features. Thus, our examples will show that we detect the same points in an image after they are rotated. In our comparison, features for our CG method are first obtained using the six largest magnitude eigenvalues at each pixel. The CG method's feature descriptor consists of the six largest eigenvalues of the Cayley adjacency, which represent the

Table 4 Number of exact feature matches for Cayley (CG), DAISY, and SIFT methods across various rotation angles on a 160×200 image

Angle	CA	DAISY	SIFT
15	69	215	79
30	27	45	31
45	19	15	16
60	23	6	11
90	33	11	19
135	27	8	6
225	29	7	5

The table highlights the superior rotation invariance of the Cayley method, which achieves the highest number of matches for angles 45° and greater despite using a smaller feature vector of size 6

decorrelated spectral information. As discussed in Sect. 2.2, the CG spectra are sparse; only the first M non-zero eigenvalues are needed, resulting in a much smaller feature vector than other methods. We also obtain features using the other methods, dense-SIFT, and DAISY, before and after applying rotation to test whether those features match after transformation. Brute force matching based on Euclidean distance between the feature descriptors is used to find matches, and the non-unique matches are removed. Once a set of corresponding coordinates is established between the original image and its transformed counterpart, the matched coordinates of the transformed image are rotated back to their original orientation. An exact match is confirmed when the orientation-restored coordinates align precisely with their initial counterparts. The total number of exact matches is then used as a metric to evaluate the effectiveness of our method in generating unique, rotation-invariant features. This approach provides a measure for assessing the quality and invariance of the proposed feature generation technique. The image we are testing is found in the Oxford dataset (Mikolajczyk et al., 2005) and is scaled to a size of 160×200.

Our comparison results can be summarized in Table 4. Here, we show the test angle in the leftmost column and the number of matches for each method. Overall, our method has the most exact matches for rotations of 45 and higher despite having a feature vector of only size 6. The DAISY method has many more matches for rotations 15 and 30, whereas our Cayley spectra have many more matches after 45 degrees of rotation. This indicates that though all methods experience a decrease in feature matches with the amount of rotation, our method experiences the lowest rate of decline, indicating better rotation invariance. At the higher rotations of 135 and 225, our CG method has around triple the number matches compared to DAISY. As expected, DAISY outperforms dense-SIFT for feature matching since it is designed to run on every pixel. Our method outperforms standard SIFT for all rotations except for the 15 and 30 degree rotations. This indicates that our feature-matching method is more robust to large amounts of rotation while using a smaller feature vector size.

We visualize the matches for all three methods, Cayley, DAISY, and SIFT, in the Figure shown in 13 for the 45-degree rotation. The figure shows that our CG features can detect pixel matches even with added rotation. We noticed that more variety and quantity of features are matched using the Cayley graph method. This is shown in the figure by a line between the matching pixels in the original and transformed images. The Cayley graph has 19 exact matches, whereas there were only 15 and 16 for DAISY and SIFT feature matching, respectively. We also note that DAISY and SIFT have less variety of feature matches, as shown by the fact that most match points are centered on a smaller portion of the image. These visual

results indicate that the CG method may account for greater details, resulting in more unique match points and increased rotation invariance.

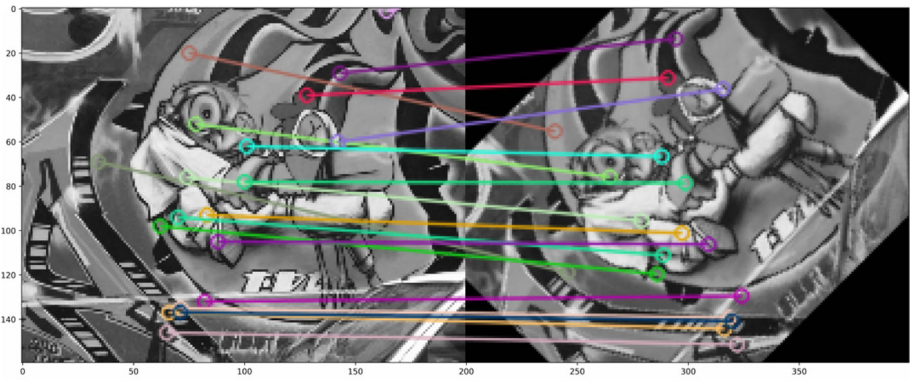
6.3 3D boundary detection

This section shows that the ratio of the first two eigenvalues of the Cayley graph can provide deeper insight into the decorrelation of signals, which results in our ability to detect 3D boundaries. The main goal is to link the Cayley graph method to the Harris corner detection and KL spectral information. The eigenvalue ratio is important because, like in the KLT, the first eigenvector is the principal component. It captures more variability, signified by the magnitude of the first eigenvalue, resulting in a larger ratio between the first and second eigenvalues than the next pair of consecutive eigenvalues. This behavior occurs because the KLT aims to find the optimal basis for decomposition, which means the ratio of the largest eigenvalue to the second largest will always be maximized for the spectra of decorrelated eigenvectors. This behavior is utilized in the Harris corner detector.

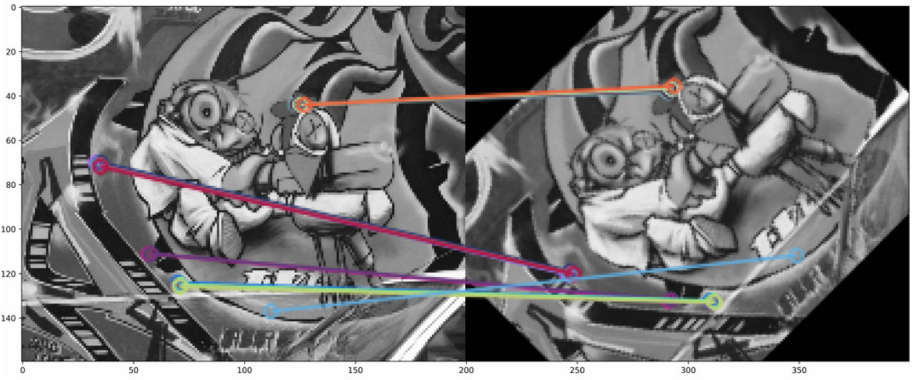
The understanding of why the eigenvalue ratios are used can be explained by describing the Harris corner detector. The Harris corner detector relies on finding image gradients, both x and y gradients or partial derivatives, I_x, I_y, at any given point of an image, which is quickly given by the corresponding Sobel x and y filter operations. We can visualize the gradients of an image at every pixel as a cluster of points that follow the shape of an ellipse with the axes that follow the I_x and I_y values. The Harris corner detection method creates a matrix from the partial derivative information of the various images, similar to the autocovariance matrix found in a KLT. These gradient patterns are directly used in the Harris corner detector by analyzing the two principal components of these 2D point cloud clusters of gradients. These principal components and their two eigenvalues can be obtained from the autocovariance matrix described in Eq. 15. Here, $w(i, j)$ represents a windowing function that determines the neighborhood of each pixel, and k is a constant parameter. The corresponding eigenvalues of the principal components give us insight into detecting corners and edges. Specifically, edge data occurs when the principal eigenvalue is much larger than the second largest $\lambda_1 >> \lambda_2$, whereas, for a corner, the eigenvalues are similar $\lambda_1 \sim \lambda_2$ and large. For a flat region, both eigenvalues are near zero $\lambda_1 \sim \lambda_2 \approx 0$. For this reason, one way to distinguish edges from flat regions is to simply take a ratio of the nonzero lambdas λ_1/λ_2.

$$\mathbf{M} = \sum_{i,j} w(i, j) \cdot \begin{bmatrix} I_x(i, j)^2 & I_x(i, j) \cdot I_y(i, j) \\ I_x(i, j) \cdot I_y(i, j) & I_y(i, j)^2 \end{bmatrix} \tag{15}$$

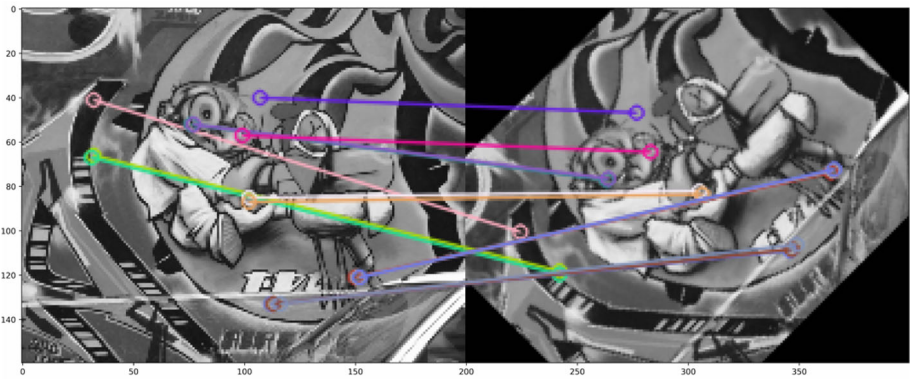
Based on this knowledge, we will leverage the eigenvalue ratio response of this Cayley graph for 3D edge detection, demonstrating a connection between Harris corner detection and autocovariance information while avoiding multiplications and derivative extraction. Though we know many efficient corner and edge detection algorithms exist (Harris and Stephens, 1988), we compare Sobel, Canny, and CG methods for 3D edge detection. These methods are chosen since they are analytical approaches with fixed calculations, in contrast to the adaptive deep learning methods. For the reasons described, we use the ratio of the two largest magnitude eigenvalues for the CG responses for edge detection. The methods are run along $3 \times 3 \times 3$ cubic volume patches. Since we are working with responses for these methods, we iteratively find the ideal threshold for the responses of all these methods across a held-out dataset.



(a) CG Matches

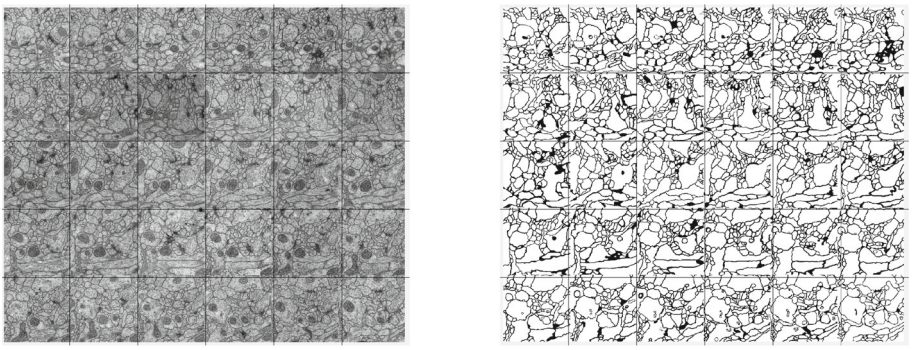


(b) SIFT Matches



(c) DAISY Matches

Fig. 13 Comparison of feature matching methods under a 45-degree image rotation. **a** CG matches, **b** SIFT matches, and **c** DAISY matches. The figure highlights the robustness of each method to rotational transformations by visualizing matched keypoints and their correspondences



(a) Original 3D Image as 30 2D scans (b) Hand-selected Ground Truths Marked

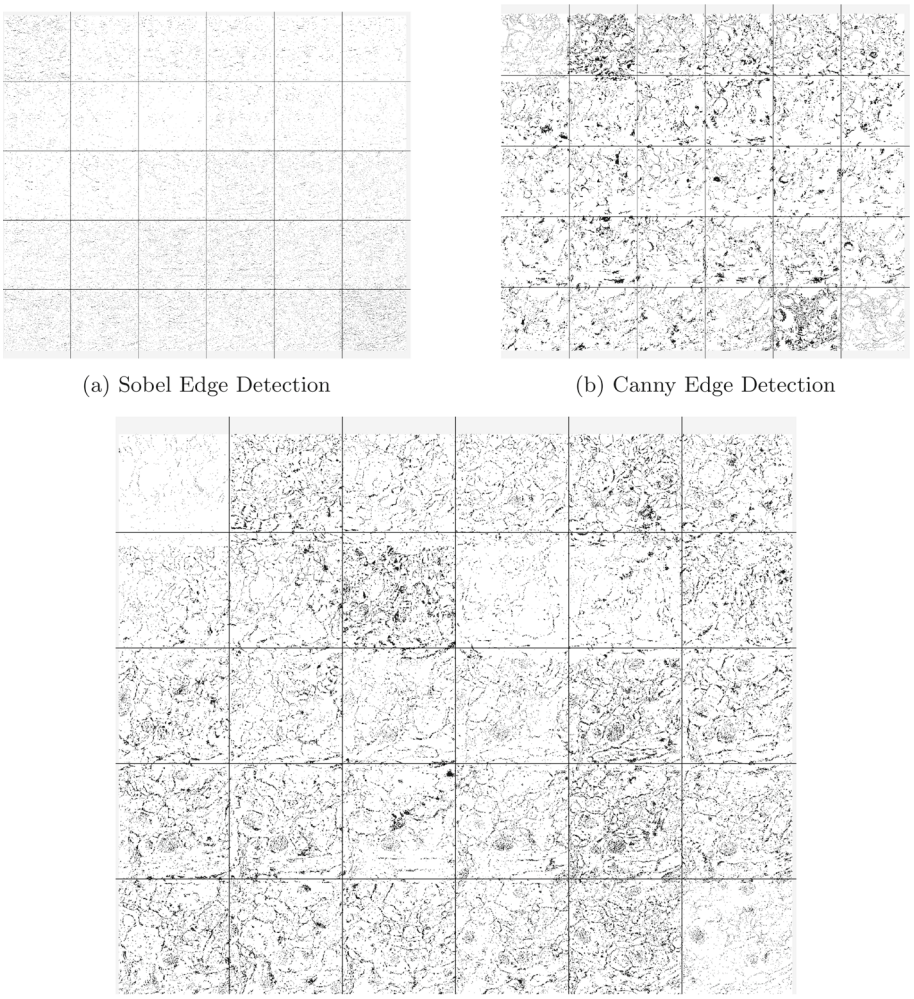
Fig. 14 Comparison of the original 3D image and the hand-selected ground truths

The metrics used for evaluation are critical, as they define the properties we aim to maximize. In our case, we know that since the edges' ground truth is created by humans, the classifier that performs best is those that find edges that a human's qualitative assessment would agree with. The metrics we use to compare our methods are the F1 score and BF score; the F1 score, also commonly referred to as the DICE score for images, can be considered the overlap of the ground truth and selected edge responses. For the BF score(Boundary F1), correct classification occurs if the edge is within a certain number of pixels away from a ground-truth edge; this gives a more reasonable error that aligns well with human-perceived results (Csurka et al., 2013). We choose a pixel distance of 3, which is fairly strict; if the edge marking is under 3 pixels away from the correct label, it is considered correct.

For this comparison, we use the three-dimensional CG method as described in Sect. 5 to detect 3D dimensional boundaries. We use a dataset from ISBI 2012 known as the connectome segmentation (Arganda-Carreras et al., 2012). For this dataset, the goal is to find the boundaries of the cell walls and highlight them. The dataset is from a single large 3D scan of $512 \times 512 \times 30$. This dataset has been used by previous segmentation methods that use Cayley tables to create 3D equivariant neural network response (Worrall and Brostow, 2018) and similarly applied it to the segmentation of the dataset. We compare the 3D versions of the Sobel filter, Canny operations (Monga et al., 1991), and CG methods, and then we compare the classification results of each pixel using the F1 and BF scores.

Figure 14a and b show the original image and ground truth. Note that the 3D image scans are visualized as 30 2D slices side by side. We hold out a small portion of $10 \times 10 \times 10$ as a segment of the image, find the responses, and use this to iteratively find the optimal threshold using the ground truths. We then ran our methods on the entire dataset to obtain the raw responses and performed the binary edge classification with our determined threshold.

The thresholded responses for the Sobel, Canny, and CG methods are shown in Fig. 15. This figure qualitatively shows that the CG method performs better than both Canny and Sobel in highlighting the fine-grained detailed edges. Also, we show that the Canny method performs much better than the Sobel method at finding complex 3D boundaries. However, it is important to note that despite these observations, none of the methods fully align with the ground truth shown in Fig. 14b when using eigenvalue ratios alone. This limitation underscores the challenge of achieving high accuracy on manually defined boundaries through analytical approaches. Advanced decision-making capabilities that better handle hand-selected bound-



(a) Sobel Edge Detection

(b) Canny Edge Detection

(c) CG Method Detection

Fig. 15 Comparative visualization of edge detection methods

ary characteristics are more effectively implemented in deep learning classifiers (Lou et al., 2021; Worrall and Brostow, 2018).

Table 5 compares our methods with the previously defined evaluation criteria. The precision, recall, F1, and BF scores of all methods are shown when compared against the hand-selected ground truth in (Arganda-Carreras et al., 2012). With metrics such as the F1 score, Canny has worse results than Sobel. However, the BF score shows that the Canny method performs better than the Sobel method and finds the edges that are proximate to the boundaries. This aligns with the fact that the BF score is more related to human-perceived results for boundary detection and seems to match our qualitative results from Fig. 15. For our CG method, results are similar to what we expected, with the CG method consistently performing better than the standard analytical methods for both F1 and BF scores.

Table 5 Performance metrics for Sobel, Canny, and CG edge detection methods, including precision, recall, F1 score, and boundary-focused (BF) score

Metric	Sobel	Canny	CG
Precision	0.7837	0.7840	0.8142
Recall	0.9755	0.9139	0.9494
F1	0.8691	0.8440	0.8766
BF score	0.4834	0.5815	0.6321

All metrics are evaluated against the ground truth

7 Conclusion

This work builds upon the work in Thornton (2005), where the connection between the spectra of the KL transform and the spectra of Cayley graphs is first made. In our research, we extend the previous method to higher dimensions. First, we introduce higher dimensional symmetry groups $(S_n)^N$ created from the direct product of groups and their corresponding Cayley graphs. Then, we use the appropriate Cayley graph generator function to extract the decorrelation information from the N-dimensional autocorrelation matrix. The properties of the resulting Cayley graph spectra have behaviors similar to those of the KL spectra, which are advantageous for minimizing the non-zero coefficients and obtaining decorrelation information. Thus, our research provides an alternative way to obtain N-dimensional decorrelated spectral information.

The realistic application of our method is then shown through comparison to standard methods for various tasks. We first applied our method to applications including signal classification and compared our CG-based methods to DFT, DCT, and mel-scaled coefficients for signal classification, which showed the CG spectra have decorrelation information that is useful and can complement other methods. We then also used higher-dimensional signals for 2D feature matching and compared them against methods such as SIFT and DAISY. Our CG method ended up having more feature matches with higher amounts of rotation invariance. For a third example, 3D edge detection, our method performs better than the standard analytical methods of 3D Sobel and 3D Canny methods for many metrics. While our method demonstrates computational advantages over autocovariance-based approaches, convolution-based methods are generally more efficient and widely adopted for realistic applications. Future work will focus on optimizing our approach to further reduce computational complexity and improve its practical applicability. Overall, this research extends the connection between KL transform and Cayley graph spectra to higher dimensions and provides evidence of its practical value across multiple tasks.

Author contributions A.S. Wrote the main text and implemented experiments. D. Y. Designed experiments and implemented experiments. Guided writing. E. L. Guided analysis and experimental setup. M.T. Guided theory and original project description and setup. All authors reviewed the manuscript.

Funding Open access funding provided by SCELC, Statewide California Electronic Library Consortium

Data availability No datasets were generated or analysed during the current study.

References

Al-Asad, J.F., Moghadamjoo, A., & Ying, L. (2009). Ultrasound image de-noising through Karhunen-Loeve (K-L) transformwith overlapping segments. In *2009 IEEE international symposium on biomedical imaging: from nano to macro* (pp. 318–321). https://doi.org/10.1109/ISBI.2009.5193048. https://ieeexplore.ieee.org/abstract/document/5193048 Accessed 2024-07-29

Arganda-Carreras, I., Seung, H. S., Cardona, A., & Schindelin, J. (2012). Segmentation of neuronal structures in em stacks challenge - ISBI 2012. In *Proceedings of the international symposium on biomedical imaging (ISBI)*http://tinyurl.com/d2fgh7g

Beck, K., Ghandehari, M., Hudson, S., & Paltenstein, J. (2024). Frames for signal processing on Cayley graphs. *Journal of Fourier Analysis and Applications, 30*(6), 66.

Bernasconi, A., & Codenotti, B. (1999). Spectral analysis of Boolean functions as a graph eigenvalue problem. *IEEE Transactions on Computers, 48*(3), 345–351. https://doi.org/10.1109/12.755000.

Bientinesi, P., Dhillon, I. S., & Van De Geijn, R. A. (2005). A parallel eigensolver for dense symmetric matrices based on multiple relatively robust representations. *SIAM Journal on Scientific Computing, 27*(1), 43–66.

Brunelli, R. (2009). *Template matching techniques in computer vision: Theory and practice*. Wiley.

Csurka, G., Larlus, D., Perronnin, F., & Meylan, F. (2013). What is a good evaluation measure for semantic segmentation?. In *Bmvc*, (vol. 27, pp. 10–5244). Bristol.

Gallian, J. (2021). *Contemporary abstract algebra* (9th ed.) Chapman and Hall/CRC.

Harris, C., & Stephens, M. (1988) A combined corner and edge detector. In *Alvey vision conference*, (vol. 15, pp. 10–5244). Citeseer.

Juan, L., & Gwun, O. (2009). A comparison of sift, pca-sift and surf. *International Journal of Image Processing (IJIP), 3*(4), 143–152.

Karhunen, K. (1946). Zur spektraltheorie stochastischer prozesse. *Ann. Acad. Sci. Fennicae, AI* 34

Kittler, J., & Young, P. C. (1973). A new approach to feature selection based on the Karhunen-Loeve expansion. *Pattern Recognition, 5*(4), 335–352. https://doi.org/10.1016/0031-3203(73)90025-3

Kok, K., & Rajendran, P. (2018). Validation of Harris detector and eigen features detector. *IOP Conference Series: Materials Science and Engineering, 370*, Article 012013.

Lacouture-Parodi, Y., Habets, E.A.P., Chen, J., & Benesty, J. (2014). Multichannel noise reduction in the Karhunen-Loève expansion domain. In *IEEE/ACM transactions on audio, speech, and language processing. conference name: IEEE/ACM transactions on audio, speech, and language processing* (vol. 22(5), 9, pp. 23–936). Accessed 2024-07-29

Lehoucq, R. B., Sorensen, D. C., & Yang, C. (1998). *ARPACK users' guide: Solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM.

Li, L., & Thornton, M. A. (2005). Discrete function KL Spectrum computation over symmetry groups of arbitrary size. In *Proceedings of the international symposium on representations and methodology of future computing technologies (formerly, Reed-Muller workshop, RMW05)* (pp. 110–113).

Liu, Z., Liu, Z., & Peng, Y. (2017). Dimension reduction of Karhunen-Loeve expansion for simulation of stochastic processes. *Journal of Sound and Vibration, 408*, 168–189. https://doi.org/10.1016/j.jsv.2017.07.016

Lou, A., Guan, S., & Loew, M. (2021). Dc-unet: rethinking the u-net architecture with dual channel efficient CNN for medical image segmentation. *Medical imaging 2021: Image processing* (Vol. 11596, pp. 758–768) SPIE.

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., & Gool, L. V. (2005). A comparison of affine region detectors. *International Journal of Computer Vision, 65*, 43–72.

Monga, O., Deriche, R., Malandain, G., & Cocquerez, J. P. (1991). Recursive filtering and edge tracking: Two primary tools for 3d edge detection. *Image and Vision Computing, 9*(4), 203–214.

Moody, G. B., & Mark, R. G. (2001). The impact of the mit-bih arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine, 20*(3), 45–50.

Narang, S.K., & Ortega, A. (2011). Downsampling graphs using spectral theory. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, (pp. 4208–4211). https://doi.org/10.1109/ICASSP.2011.5947281

Ng, A., Jordan, M., & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 14.

Olmos, S., Millan, M., Garcia, J., & Laguna, P. (1996). ECG data compression with the Karhunen-Loeve transform. In *Computers in Cardiology 1996*, (pp. 253–256) https://doi.org/10.1109/CIC.1996.542521. https://ieeexplore.ieee.org/abstract/document/542521 Accessed 2024-07-29

Sandryhaila, A., & Moura, J.M.F. (2013). Discrete signal processing on graphs: Graph Fourier transform. In *2013 IEEE international conference on acoustics, speech and signal processing*, (pp. 6167–6170). https://doi.org/10.1109/ICASSP.2013.6638850

Shafipour, R., Khodabakhsh, A., Mateos, G., & Nikolova, E. (2018). Digraph Fourier transform via spectral dispersion minimization. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, (pp. 6284–6288). https://doi.org/10.1109/ICASSP.2018.8461875

Thornton, M.A. (2005). The Karhunen-Loéve Transform of discrete MVL functions. In *35th international symposium on multiple-valued logic (ISMVL'05)*, (pp. 194–199). IEEE, Calgary, Canada. https://doi.org/10.1109/ISMVL.2005.48. http://ieeexplore.ieee.org/document/1423181/ Accessed 2024-02-21

Thornton, M.A., Miller, D.M., & Townsend, W.J. (2002). Chrestenson spectrum computation using Cayley color graphs. In *Proceedings 32nd IEEE International Symposium on Multiple-Valued Logic*, (pp. 123–128). https://doi.org/10.1109/ISMVL.2002.1011079

Tola, E., Lepetit, V., & Fua, P. (2009). Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 32*(5), 815–830.

Townsend, W. J., & Thornton, M. A. (2001). Walsh spectrum computations using Cayley graphs. In *Proceedings of the 44th IEEE 2001 midwest symposium on circuits and systems. MWSCAS 2001 (Cat. No. 01CH37257)* (Vol. 1, pp. 110–113). IEEE.

Warden, P. (2018). Speech commands: A dataset for limited-vocabulary speech recognition arXiv:1804.03209 arXiv preprint.

Worrall, D., & Brostow, G. (2018). *CubeNet: Equivariance to 3D rotation and translation* (pp. 567–584).