Efficient Eigen-Decomposition for Low-Rank Symmetric Matrices in Graph Signal Processing: An Incremental Approach

Qinwen Deng*, *Member, IEEE,* Yangwen Zhang*, Mo Li, Songyang Zhang, *Member, IEEE* and Zhi Ding, *Fellow, IEEE*

***Abstract*—Graph spectral analysis has emerged as an important tool to extract underlying structures among data samples. Central to graph signal processing (GSP) and graph neural networks (GNN), graph spectrum is often derived via eigen-decomposition (ED) of graph representation (adjacency/Laplacian) matrix. Many real-world applications feature dynamic graphs whose representation matrix size varies over time. Such evolving graph usually shares part of the same structures with the previous graphs. We consider efficient ways to estimate the K dominant eigenvectors of the graph representation matrix. We focus on an iterative ED algorithm for low-rank symmetric matrices to update the top K eigen-pairs of the representation matrix for a graph with increasing node size. To accommodate the growing graph size, we propose two Incremental ED algorithms for Low Rank symmetric matrices (ILRED) based on an iterative eigen-updating strategy. We also provide analysis on the resulting error performance, computational complexity and memory usage to showcase the efficiency of ILRED. The experimental results in both synthetic and real-world datasets with the context of spectral clustering and graph filtering validate the power of the proposed ILRED algorithms.**

***Index Terms*—eigen-decomposition, eigen-updating, graph signal processing, graph spectral analysis**

I. Introduction

GRAPH spectral analysis has recently become an important tool for structural data analysis in graph signal processing (GSP) and graph neural networks (GNN). This surge in interest is driven by the wide-ranging application in diverse domains, such as the Internet-of-Things (IoT), social networks, financial data, traffic patterns, and biological systems [1]–[3]. All these network-structured data can be naturally modeled by graphs, which highlights the significance of employing graph-based methods for signal processing. In GSP, the graph spectrum, also known as graph Fourier basis is derived from eigen-decomposition (ED) of the representation matrix (adjacency/Laplacian) of the graph, based on which the graph spectral approaches, such as graph spectral analysis and filtering [4], [5], can be designed for data analysis. To derive the graph Fourier space for efficient structural data processing, a fast and accurate ED has become prevalent within GSP.

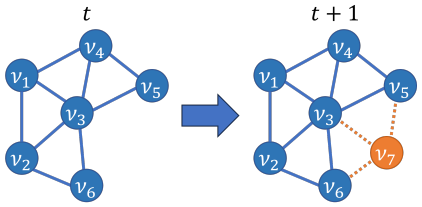

Fig. 1. Example of an incrementally-updated graph at time t and time $t+1$. The additional vertices and edges at time $t+1$ are labeled in orange and orange dash lines, respectively.

The pursuit of efficient ED algorithms with low computational complexity has attracted intensive investigation in GSP. The high computational complexity and heavy peak memory utilization of ED lead to significant bottleneck in graph spectral analysis for large-sized datasets. Despite the achievements of many existing works on efficient approximation of eigen-pairs for all scales of matrices [6], most algorithms focus on static graphs. However, in many realistic applications, such as dynamic point clouds and social networks, the graph size keeps changing over time. When new data samples come, the node size increases and the graph structure partially updates. For example, as shown in Fig. 1, there is one additional vertex at time $t+1$, which increases the dimension of the corresponding representation matrix. Then, the eigen-pairs need to be re-computed for GSP spectral analysis. In these dynamic scenarios, the re-computation of eigen-pairs upon each alteration in the graph presents a significant computational burden. This demand for continual re-computation highlights the need to develop ED algorithms that can effectively handle the evolving nature of dynamic graphs. In this work, we focus on the updating problem to upon the largest K eigenvalues and the corresponding eigenvectors of symmetric adjacency matrices for undirected graphs as the graph size increases (*incrementally-updated graph*). The estimated top K eigen-pairs have broad applications for graph spectral analysis, including spectral clustering [7]–[9] and graph filtering.

Many existing eigen-updating works focus on the rank-one or rank-k ED update [10]–[12]. However, these approaches can not be directly applied in real-world graph spectral analysis, since there is no guarantee that each change of the graph will lead to a rank-one or rank-k matrix. Only a few studies focus

*These authors contributed equally to this work.

Qinwen Deng and Zhi Ding are with Department of Electrical and Computer Engineering, University of California at Davis, Davis, CA 95616 (E-mail: dcdengqinwen@gmail.com; zding@ucdavis.edu).

Yangwen Zhang and Mo Li are with Department of Mathematics, University of Louisiana at Lafayette, Lafayette, LA, 70504 (E-mail: yangwen.zhang@louisiana.edu; mo.li@louisiana.edu).

Songyang Zhang is with Department of Electrical and Computer Engineering, University of Louisiana at Lafayette, Lafayette, LA, 70504 (E-mail: songyang.zhang@louisiana.edu).

on such issues. The authors in [13], [14] proposed an eigen-pair updating algorithm based on the perturbation of matrices in a generalized eigenvalue system. This method needs to calculate the matrix inverse whenever updating the eigenvectors, which is computationally costly. An incremental eigen-approximation algorithm was proposed in [15]. However, compared with the general decomposition expression of the updated matrix given by [16], the proposed algorithm ignores the residual elements to further simplify the decomposition expression, which results in large approximation errors. This assumption also implies that the top K eigen-pairs of the original matrix shall maintain their relative order to be in the top K in the updated matrix, which may not hold in general cases. Another kind of fast eigen-function tracking algorithm was proposed in [17], where the changes of the adjacency matrices are viewed as perturbations. Based on the matrix perturbation theory in [18], the authors proposed the first-order and higher-order eigen-pair tracking algorithms, named TRIP-BASIC and TRIP, respectively. However, the approximation error of the TRIP-BASIC algorithm is large while the computational complexity of the TRIP is high.

To efficiently solve the eigen-updating problem in *incrementally-updated graphs*, where several additional nodes are added to the original graph, we propose an **I**ncremental **ED** algorithm for **L**ow **R**ank symmetric matrices, namely **ILRED**. More specifically, we propose two versions of ILRED, i.e., ILRED-BASIC and ILRED-FAST. Our approaches update top k eigen-pairs by iteratively increasing the matrix size one by one instead of updating the increase of multiple dimensions at once. This strategy provides us with two advantages. First, we have more control over the approximation errors in each iteration. Compared with the proposed algorithm in [15], we design a novel error correction branch whenever the approximation error exceeds the tolerance in each iteration. Second, the intermediate variables of our proposed algorithm have a smaller size, which reduces the peak memory usage. Consequently, our method exhibits smaller approximation errors and less peak memory usage while maintaining the same order of computational complexity. Our theoretical analysis of error performance, computational complexity, and memory cost demonstrate the accuracy and efficiency of the proposed ILRED. The experimental results further validate the robustness of ILRED in both synthetic and realistic datasets with applications to spectral clustering and graph filtering.

We summarize our contributions as follows:

- We propose two versions of ILRED, i.e., ILRED-BASIC and ILRED-FAST for graph spectrum calculation of incrementally updated graphs in GSP. To reduce the approximation errors, we design a novel error correction branch whenever the approximation error exceeds the tolerance in each iteration. To the best of our knowledge, we are the first to introduce such error correction algorithm in ED approximation.
- To gain more flexibility in error control and reduce the memory cost, we adopt an iterative eigen-updating strategy for our ILRED algorithms. In each iteration, we increase the size of the matrix by one and update the tracking eigen-pairs.
- To demonstrate the accuracy and efficiency of our proposed ILRED algorithms, we provide the theoretical analysis of the error performance and computational complexity.
- Beyond the theoretical analysis, we implement the ILRED algorithms for graph spectral analysis, such as spectral clustering and graph filtering in both synthetic and real-world datasets.

We organize the rest of the paper as follows. Following the introduction of preliminaries of graph spectral analysis and the overview of related works in Section II, we present the details of ILRED algorithms in Section III. We then analyze the error performance, computational complexity, and memory usage of our proposed ILRED algorithms in Section IV. We also present the experimental results of the proposed methods in both synthetic and real-world datasets in Section V, before summarizing our work in Section VI.

II. Preliminary and Related Works

In this section, we first briefly introduce the preliminaries of graph spectral analysis and then overview the related works.

A. Notations

For convenience, the symbols used in this work are summarized in Table. I as follows.

TABLE I
SYMBOLS USED IN THE PAPER.

Symbol	Definition and Description
$G = (\mathcal{V}, \mathcal{E})$	undirected graph
$\mathcal{V}$	set of vertices in graph
$\mathcal{E}$	set of edges in graph
v_i	the i-th vertex in graph
$\mathbf{G}$	representation matrix of graph
$\mathbf{A}$	adjacency matrix of graph
$\mathbf{L}$	Laplacian matrix of graph
$\mathbf{s}$	graph signal of length N
$\mathbf{Q}$	orthonormal matrix whose columns are the eigenvectors
$\mathbf{\Sigma}$	diagonal matrix with the eigenvalues on the diagonal
$\mathbf{M}_{\text{old}}$	initial matrix with known eigen-decomposition result
$\mathbf{M}_{\text{new}}$	final matrix with eigen-decomposition result to be updated

B. Graph Spectral Analysis

Graph signal processing (GSP) has recently emerged as an important tool for structural data analysis due to its power in capturing underlying data correlations [1]. Consider an undirected graph $G = (\mathcal{V}, \mathcal{E})$ with N vertices, where $\mathcal{V}$ denotes the set of vertices, i.e., $\mathcal{V} = \{v_1, \cdots, v_N\}$, and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ represents the edges. A representation matrix $\mathbf{G} \in \mathbb{R}^{N \times N}$ can be used to describe the geometric structure of the graph G. Commonly used representation matrices include adjacency matrix $\mathbf{A}$ or Laplacian matrix $\mathbf{L}$, which are symmetric for undirected graphs and usually sparse in large datasets. Graph signals are the attributes of vertices, which can be written as vector $\mathbf{s} = [s_1, s_2, \cdots, s_N]^\top \in \mathbb{R}^N$.

Real-world datasets, such as point clouds, sensor data, and images, can be naturally represented by graphs. Given the data points, the construction of the representation matrices

of graphs plays a pivotal role in capturing the underlying topology of the dataset [2]. Among the various methods for graph construction, a notable approach is model-based graph construction. This technique leverages domain knowledge to construct graphs using models, such as ϵ-neighborhood graph, tailored to the dataset at hand. In an ϵ-neighborhood graph, two vertices are connected by edge if their Euclidean distance is smaller than a given threshold. In this work, we build the ϵ-neighborhood graph based on the intrinsic resolution d_r of the dataset and the Gaussian kernel. The edge weight between the i-th and j-th vertex is

$$\mathbf{A}_{i,j} = \begin{cases} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{d_r^2}} & \|\mathbf{x}_i - \mathbf{x}_j\|^2 \leq \epsilon \\ 0 & \text{otherwise} \end{cases}, \tag{II.1}$$

where $\mathbf{x}_i$ is the vector of data attributes for the i-th vertex. The intrinsic resolution d_r of the dataset is defined as the mean of the smallest distance from each vertex to all other vertices in the dataset, i.e., for a dataset with N vertices, $d_r = \frac{1}{N}\sum_{i=1,i\neq j}^{N} \min_j \|\mathbf{x}_i - \mathbf{x}_j\|$.

The graph spectral space, also referred to as the graph Fourier space, is defined based on the eigenspace of the representing matrix $\mathbf{G}$. Suppose that the eigen-decomposition of $\mathbf{G}$ is $\mathbf{G} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}$. Here, $\mathbf{Q} \in \mathbb{R}^{N\times N}$ denotes the orthonormal matrix whose columns are the eigenvectors of $\mathbf{G}$, and $\mathbf{\Lambda} \in \mathbb{R}^{N\times N}$ is a diagonal matrix with the eigenvalues of $\mathbf{G}$ on the diagonal, i.e., $\mathbf{\Lambda} = \text{diag}(\lambda_1, \cdots, \lambda_N)$. Then the graph Fourier transform (GFT) is defined as

$$\hat{\mathbf{s}} = \mathbf{Q}^{-1}\mathbf{s}, \tag{II.2}$$

whereas the inverse GFT is given by $\mathbf{s} = \mathbf{Q}\hat{\mathbf{s}}$.

From the definitions of GFT, the concept of graph spectral analysis was developed as an important tool for signal processing and data analysis. Graph frequency analysis and graph Fourier filter design were introduced in [4], [5]. The basic non-trivial filter defined on the graph G is called the graph shift, which is defined as

$$\tilde{\mathbf{s}} = \mathbf{A}\mathbf{s}. \tag{II.3}$$

Additionally, the linear, shift-invariant graph filters can be expressed as the polynomials of the adjacency matrix $\mathbf{A}$, which can be written as

$$h(\mathbf{A}) = h_0\mathbf{I} + h_0\mathbf{A} + \cdots + h_L\mathbf{A}^L. \tag{II.4}$$

The output signal can be expressed as

$$\tilde{\mathbf{s}} = \mathbf{H}(\mathbf{s}) = h(\mathbf{A})\mathbf{s}. \tag{II.5}$$

By using GFT, the Fourier transform coefficients of the filtered signal $\tilde{\mathbf{s}}$ can be written as

$$\begin{aligned} \mathbf{Q}^{-1}\tilde{\mathbf{s}} &= \mathbf{Q}^{-1}h(\mathbf{A})\mathbf{Q}\hat{\mathbf{s}} \\ &= \begin{bmatrix} h(\lambda_1) & & \\ & \ddots & \\ & & h(\lambda_N) \end{bmatrix}\hat{\mathbf{s}}, \end{aligned} \tag{II.6}$$

which indicates the design of the graph spectral filter.

There are two kinds of commonly used graph filters: ideal graph filter and Haar-like graph filter. The expression of $h(\mathbf{A})$ for ideal graph filter is

$$h(\mathbf{A}) = diag(1, \cdots, 1, 0, \cdots, 0). \tag{II.7}$$

The number of ones for ideal graph filter can be determined by the threshold of the eigenvalues. On the other hand, the $h(\mathbf{A})$ for Haar-like graph filter can be written as

$$h(\mathbf{A}) = \mathbf{I} - \mathbf{\Lambda}_{norm}, \tag{II.8}$$

where $\mathbf{I}$ represents the identity matrix of size N by N and $\mathbf{\Lambda}_{norm}$ is the normalized eigenvalue matrix by normalizing the largest eigenvalue of $\mathbf{A}$ to 1.

Graph frequency analysis has been applied to many practical applications. The ideal low-pass and high-pass graph Fourier filters were used in [19] in the anomaly detection algorithms for wireless sensor networks. The authors in [20] used graph frequency analysis to identify anatomy-aligned function signals in the brain and uncover an integrated structure-function relation of human behavior. More potentials of graph frequency analysis and graph Fourier filtering in structure-informed study of functional brain dynamics were revealed in [21]. Another example is the spectral clustering methods based on low-frequency eigenvectors of the Laplacian matrix [22], which was widely used as a benchmark clustering method.

Within the scope of updating the spectral clustering results for dynamic graphs, existing works can be divided into two categories. The first approach is to iteratively update the ED result, then update the clustering result based on the approximated eigenvectors, which will be reviewed in Section II-C. The second approach is to estimate the clustering result based on the representative sets, such as the method proposed in [23]. This method instantly assigns cluster labels to newly added nodes based on the representative reliability of every node in each cluster. After that, the eigenvalues are also updated to estimate the number of clusters. However, when the estimated number of clusters changes, this method needs to re-initialize the algorithm by clustering based on the ED of the new graph, which increases the computational complexity.

C. Fast Matrix Decomposition

Many existing works of ED updating focus on rank-one or rank-k ED update problems [10]–[12]. These approaches can not be directly applied in real-world graph based applications since there is no guarantee that each change of the graph has to be limited to rank-one or rank-k. Only a few studies are focused on such issues. The authors in [13], [14] considered a generalized eigenvalue system such that $\mathbf{B}\mathbf{x} = \lambda\mathbf{C}\mathbf{x}$, where both $\mathbf{B} \in \mathbb{R}^{N\times N}$ and $\mathbf{C} \in \mathbb{R}^{N\times N}$ are symmetric, and λ and $\mathbf{x}$ are the eigenvalue and corresponding eigenvector that needs to be determined. They proposed the eigen-pair updating algorithm based on finding the perturbation of eigenvalue and eigenvectors according to the perturbation of matrices $\mathbf{B}$ and $\mathbf{C}$. However, this method needs to calculate the matrix inverse whenever updating the eigenvectors, which is computationally costly. An efficient incremental eigen-approximation algorithm

was proposed in [15]. This algorithm assumes that the updated matrix $\mathbf{M}_{\text{new}} \in \mathbb{R}^{(N+1)\times(N+1)}$ can be written in the form that

$$\begin{aligned}\mathbf{M}_{\text{new}} &= [\mathbf{B} \quad \mathbf{c}]^\top [\mathbf{B} \quad \mathbf{c}] \\ &= \begin{bmatrix} \mathbf{B}^\top\mathbf{B} & \mathbf{B}^\top\mathbf{c} \\ \mathbf{c}^\top\mathbf{B} & \mathbf{c}^\top\mathbf{c} \end{bmatrix},\end{aligned} \tag{II.9}$$

where $\mathbf{M}_{\text{old}} = \mathbf{B}^\top\mathbf{B} \in \mathbb{R}^{N\times N}$ is the original matrix before the size change, $\mathbf{B} \in \mathbb{R}^{K\times N}$ is the decomposition matrix of $\mathbf{M}_{\text{old}}$, $\mathbf{c} \in \mathbb{R}^{K\times 1}$ is the difference between the decomposition matrix of $\mathbf{M}_{\text{new}}$ and $\mathbf{M}_{\text{old}}$, and K is the highest rank of $\mathbf{M}_{\text{old}}$ and $\mathbf{M}_{\text{new}}$. However, this decomposition expression is incomplete. The general expression for the decomposition of the updated matrix, given by [16], should be

$$\begin{aligned}\mathbf{M}_{\text{new}} &= \begin{bmatrix} \mathbf{B} & \mathbf{c_1} \\ 0 & c_2 \end{bmatrix}^\top \begin{bmatrix} \mathbf{B} & \mathbf{c_1} \\ 0 & c_2 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{B}^\top\mathbf{B} & \mathbf{B}^\top\mathbf{c_1} \\ \mathbf{c_1}^\top\mathbf{B} & \mathbf{c_1}^\top\mathbf{c_1} + {c_2}^2 \end{bmatrix},\end{aligned} \tag{II.10}$$

where $\mathbf{c_1} \in \mathbb{R}^{K\times 1}$ and $c_2 \in \mathbb{R}$. Compare Eq. II.9 and Eq. II.10, the former decomposition expression drops the residual element of ${c_2}^2$, which increases the approximation error. Additionally, the former decomposition expression overlooks the existence of additional rows in the decomposition matrix. These rows could correspond to the new top K eigen-pairs of the updated matrix $\mathbf{M}_{\text{new}}$, and be kept in the final result. Therefore, the approximation errors of the proposed algorithm is enlarged.

Other kinds of fast eigen-function tracking algorithms were proposed in [17], where the changes of the adjacency matrices are viewed as the perturbation. Based on the matrix perturbation theory in [18], the authors proposed first order and higher order eigen-pairs tracking algorithms, named TRIP-BASIC and TRIP, respectively. However, the approximation error of the TRIP-BASIC algorithm is in the order of the norm of the perturbation of adjacency matrix. The approximation error will be high when the norm of perturbation is large and there is no control on that. On the other hand, the computational complexity of the TRIP is $O(K^4)$ for each iteration, where K is the number of eigenvalues that are tracked. Such complexity is very high for a large K.

III. Method and Analysis

In this section, we introduce our incremental ED algorithm for low-rank symmetric matrices (ILRED). In many big data analysis, the graph is sparse and the representation matrix are usually low-rank. We assume that the input matrix of the algorithm is low-rank and positive semi-definite in this section for better understanding while keeping the mathematical integrity of the expression.

A. Basic Incremental Eigen-Decomposition

We first introduce one iteration of our basic version of ILRED algorithm, i.e., ILRED-BASIC, which only increases the matrix size by one and updates the eigen-pairs. In practical applications where we need to increase the matrix size by ℓ from time t to time $t+1$, we could run the following iteration

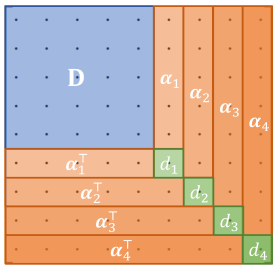

Fig. 2. Example of iterative update of our ILRED algorithm. $\mathbf{M}_{\text{old}}$ is the original low-rank symmetric matrix whose ED matrices are known. In each iteration of the ILRED algorithm, one new pair of $\boldsymbol{\alpha}_i$ and d_i are used as the inputs of the algorithm to update the ED result.

by ℓ times, as shown in Fig. 2. In each iteration, one new pair of $\boldsymbol{\alpha}_i$ and d_i are used as the inputs of the algorithm to update the ED result. In this way, we will have better control on the approximation errors in each iteration and lower peak memory usage.

Assuming we already have the truncated eigen-decomposition of rank k for the matrix $\mathbf{M}_{\text{old}} \in \mathbb{R}^{n\times n}$ as follows:

$$\mathbf{M}_{\text{old}} = \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top, \tag{III.1}$$

where $\mathbf{\Sigma} \in \mathbb{R}^{k\times k}$ is a diagonal matrix with the k ordered eigen values of $\mathbf{M}_{\text{old}}$ on the diagonal, i.e., $\mathbf{\Sigma} = diag(\lambda_1, \cdots, \lambda_k)$, $\mathbf{Q} \in \mathbb{R}^{n\times k}$ is the matrix of the corresponding k eigenvectors of $\mathbf{M}_{\text{old}} \in \mathbb{R}^{n\times n}$, with $\mathbf{Q}^\top\mathbf{Q} = I_k$,

Our goal is to update eigen-decomposition results by only employing $\mathbf{Q} \in \mathbb{R}^{n\times k}$, additional non-diagonal vector $\boldsymbol{\alpha} \in \mathbb{R}^n$, and additional diagonal element $d \in \mathbb{R}$ for the following incrementally-updated graph representation matrix $\mathbf{M}_{\text{new}}$, i.e.,

$$\mathbf{M}_{\text{new}} = \begin{bmatrix} \mathbf{M}_{\text{old}} & \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^\top & d \end{bmatrix}. \tag{III.2}$$

We compute the residual vector $\mathbf{e}$ of $\boldsymbol{\alpha} \in \mathbb{R}^n$ by projecting it onto the subspace spanned by the columns of $\mathbf{Q} \in \mathbb{R}^{n\times k}$, i.e.,

$$\mathbf{e} = \boldsymbol{\alpha} - \mathbf{Q}\mathbf{Q}^\top\boldsymbol{\alpha}. \tag{III.3}$$

The norm of the residual vector $\mathbf{e}$ is corresponding to the approximation error of the final result. Therefore, we want to correct the errors when it is too large. On the other hand, when the approximation error is small, there is no need to make such a correction with additional complexity cost. In our algorithm, we use a predefined tolerance ε to determine whether the approximation error is small or not.

1) When the norm of the residual vector is small: If the norm of the residual vector is small, i.e., $p = \|\mathbf{e}\| \leq \varepsilon$, then we perform the p-truncation by

$$\boldsymbol{\alpha} \approx \mathbf{Q}\mathbf{Q}^\top\boldsymbol{\alpha}, \tag{III.4}$$

so that the matrix $\mathbf{M}_{\text{new}}$ can be approximated by

$$\begin{aligned}\mathbf{M}_{\text{new}} &\approx \begin{bmatrix} \mathbf{M}_{\text{old}} & \mathbf{Q}\mathbf{Q}^\top\boldsymbol{\alpha} \\ \boldsymbol{\alpha}^\top\mathbf{Q}\mathbf{Q}^\top & d \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{Q} & 0 \\ 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} \boldsymbol{\Sigma} & \mathbf{Q}^\top\boldsymbol{\alpha} \\ \boldsymbol{\alpha}^\top\mathbf{Q} & d \end{bmatrix}}_{\mathbf{Y}_1} \begin{bmatrix} \mathbf{Q} & 0 \\ 0 & 1 \end{bmatrix}^\top \\ &= \begin{bmatrix} \mathbf{Q} & 0 \\ 0 & 1 \end{bmatrix} \mathbf{Q}_1\boldsymbol{\Sigma}_1\mathbf{Q}_1^\top \begin{bmatrix} \mathbf{Q} & 0 \\ 0 & 1 \end{bmatrix}^\top. \end{aligned} \tag{III.5}$$

Here $\mathbf{Q}_1\boldsymbol{\Sigma}_1\mathbf{Q}_1^\top$ is the ED of the middle matrix $\mathbf{Y}_1$. $\mathbf{Q}_1$ and $\boldsymbol{\Sigma}_1$ can be computed by standard ED algorithm. Given that the size of $\mathbf{Y}_1$ is $(k+1)\times(k+1)$ and $k \ll n$, we can greatly reduce the computational cost compared with direct recomputing the ED of $\mathbf{M}_{\text{new}}$.

To further reduce the computational complexity as well as the memory cost, we can perform truncation by ignoring the smallest eigenvalue of $\mathbf{Y}_1$ when it is small. In our algorithm, we use another predefined threshold ε_λ to determine if the eigenvalue is small. A typical value of ε_λ is 0.1ε. If the smallest eigenvalue of $\mathbf{Y}_1$ is small, then we perform eigenvalue truncation and this suggests the following update:

$$\begin{aligned}\mathbf{Q} &\leftarrow \begin{bmatrix} \mathbf{Q} & 0 \\ 0 & 1 \end{bmatrix} \mathbf{Q}_1(:, 1:k), \\ \boldsymbol{\Sigma} &\leftarrow \boldsymbol{\Sigma}_1(1:k, 1:k). \end{aligned} \tag{III.6}$$

Otherwise, we consider the following full-dimension update

$$\begin{aligned}\mathbf{Q} &\leftarrow \begin{bmatrix} \mathbf{Q} & 0 \\ 0 & 1 \end{bmatrix} \mathbf{Q}_1, \\ \boldsymbol{\Sigma} &\leftarrow \boldsymbol{\Sigma}_1. \end{aligned} \tag{III.7}$$

The updated $\mathbf{Q}$ and $\boldsymbol{\Sigma}$ will be the input of the next iteration of the ILRED algorithm as the approximated ED of $\mathbf{M}_{\text{new}}$.

2) When the norm of the residual vector is large: If the norm of the residual vector is large, i.e., $p = \|\mathbf{e}\| > \varepsilon$, we shall not use the p-truncation. Instead, we have the fundamental theorem as follows.

Theorem 1. Let $\widetilde{\mathbf{e}} = \mathbf{e}/p \in \mathbb{R}^n$. The updated matrix $\mathbf{M}_{\text{new}}$ can be expressed by the identity

$$\begin{aligned}\mathbf{M}_{\text{new}} &= \begin{bmatrix} \mathbf{Q} & \widetilde{\mathbf{e}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} \boldsymbol{\Sigma} & 0 & \mathbf{Q}^\top\boldsymbol{\alpha} \\ 0 & 0 & p \\ \boldsymbol{\alpha}^\top\mathbf{Q} & p & d \end{bmatrix}}_{\mathbf{Y}_2} \begin{bmatrix} \mathbf{Q} & \widetilde{\mathbf{e}} & 0 \\ 0 & 0 & 1 \end{bmatrix}^\top \\ &= \begin{bmatrix} \mathbf{Q} & \widetilde{\mathbf{e}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{Q}_2\boldsymbol{\Sigma}_2\mathbf{Q}_2^\top \begin{bmatrix} \mathbf{Q} & \widetilde{\mathbf{e}} & 0 \\ 0 & 0 & 1 \end{bmatrix}^\top \end{aligned} \tag{III.8}$$

Proof. Please see Appendix A for the proof. □

It is easy to see that 0 must be an eigenvalue of $\mathbf{Y}_2$, hence the last column of $\mathbf{Q}_2$ represents an unused subspace dimension and should be suppressed. Similar to the case when the norm of the residual vector is small, when the second smallest eigenvalue of $\mathbf{Y}_2$ is small, the eigenvalue truncation update is

$$\begin{aligned}\mathbf{Q} &\leftarrow \begin{bmatrix} \mathbf{Q} & \widetilde{\mathbf{e}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{Q}_2(:, 1:k), \\ \boldsymbol{\Sigma} &\leftarrow \boldsymbol{\Sigma}_2(1:k, 1:k). \end{aligned} \tag{III.9}$$

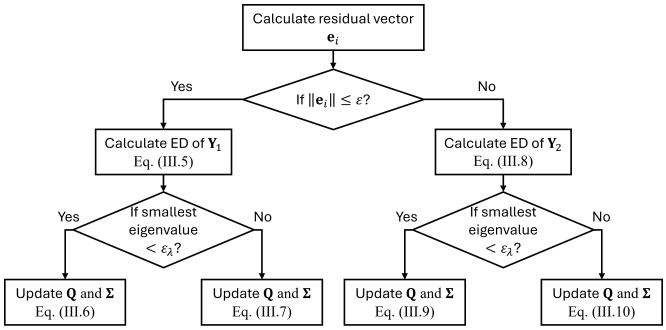

Fig. 3. Flow diagram of one iteration in algorithm 1.

Algorithm 1 Basic Incremental Eigen-Decomposition Algorithm (ILRED-BASIC)

Input: Decomposition matrix $\mathbf{Q}_{\text{init}}$, diagonal eigenvalue matrix $\boldsymbol{\Sigma}_{\text{init}}$, additional vectors $\{\boldsymbol{\alpha}_i, i \in 1, \cdots, \ell\}$, additional singular values $\{d_i, i \in 1, \cdots, \ell\}$ tolerance ε for residual vector and threshold ε_λ for eigenvalue;
1. Set $\mathbf{Q}$ and $\boldsymbol{\Sigma}$ as $\mathbf{Q}_{\text{init}}$ and $\boldsymbol{\Sigma}_{\text{init}}$ respectively;
for i from 1 to ℓ **do**
  2. Calculate the residual vector $\mathbf{e}_i$ and its norm using Eq. (III.3);
  if $p = \|\mathbf{e}\| \le \varepsilon$ **then**
    3. Calculate the ED of $\mathbf{Y}_1$ defined in Eq. (III.5);
    if the smallest eigenvalue of $\mathbf{Y}_1$ is smaller than ε_λ **then**
      4. Update $\mathbf{Q}$ and $\boldsymbol{\Sigma}$ using Eq. (III.6);
    else
      5. Update $\mathbf{Q}$ and $\boldsymbol{\Sigma}$ using Eq. (III.7);
    end if
  else
    6. Calculate the ED of $\mathbf{Y}_2$ defined in (III.8);
    if the second smallest eigenvalue of $\mathbf{Y}_2$ is smaller than ε_λ **then**
      7. Update $\mathbf{Q}$ and $\boldsymbol{\Sigma}$ using Eq. (III.9);
    else
      8. Update $\mathbf{Q}$ and $\boldsymbol{\Sigma}$ using Eq. (III.10);
    end if
  end if
end for
Output: Decomposition matrix $\mathbf{Q}$ and diagonal eigenvalue matrix $\boldsymbol{\Sigma}$ as the approximated ED of $\mathbf{M}_{\text{final}}$.

Otherwise, we perform the full-dimensional update as follows.

$$\begin{aligned}\mathbf{Q} &\leftarrow \begin{bmatrix} \mathbf{Q} & \widetilde{\mathbf{e}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{Q}_2(:, 1:k+1), \\ \boldsymbol{\Sigma} &\leftarrow \boldsymbol{\Sigma}_2(1:k+1, 1:k+1). \end{aligned} \tag{III.10}$$

The flow diagram of one update iterations is shown in Fig. 3.

For the case of multiple dimension increase, we will update the ED result by multiple iterations using the introduced algorithm. Assuming we have the ED matrices $\mathbf{Q}_{\text{init}} \in \mathbb{R}^{n\times k}$ and $\boldsymbol{\Sigma}_{\text{init}} \in \mathbb{R}^{k\times k}$ for the rank k initial matrix $\mathbf{M}_{\text{init}} \in \mathbb{R}^{n\times n}$, the final matrix $\mathbf{M}_{\text{final}} \in \mathbb{R}^{(n+\ell)\times(n+\ell)}$, where $\ell \ge 1$, we summarize our proposed basic ILRED algorithm in Alg. 1.

B. Fast Incremental Eigen-Decomposition

To reduce the computational complexity of the basic ILRED algorithm, we introduce ILRED-FAST in this section. To do this, we adopt a new decomposition model. Instead of performing rotations on the large eigenvector matrices, we maintain the eigen-decomposition of Eq. (III.1) in the form of matrix product

$$\mathbf{M}_{\text{old}} = \widetilde{\mathbf{Q}}\widehat{\mathbf{Q}}\boldsymbol{\Sigma}\widehat{\mathbf{Q}}^{\top}\widetilde{\mathbf{Q}}^{\top}, \tag{III.11}$$

where $\widetilde{\mathbf{Q}} \in \mathbb{R}^{n\times k}$ and $\widehat{\mathbf{Q}} \in \mathbb{R}^{k\times k}$ such that the product of these two matrices $\mathbf{Q} = \widetilde{\mathbf{Q}}\widehat{\mathbf{Q}}$ is orthonormal. The expansive outer matrices $\widetilde{\mathbf{Q}}$ and $\widetilde{\mathbf{Q}}^{\top}$ solely capture the eigen subspace's span. They are extended by adding rows to $\widetilde{\mathbf{Q}}$ when the rank of the update matrix remains the same, and by appending both new rows and columns when the rank of the update matrix increases. The transforms of these subspace bases to make $\boldsymbol{\Sigma}$ diagonal are maintained in a much smaller $\widehat{\mathbf{Q}}$ matrix, whose size only extends when the rank of the update matrix increases. This makes the update much faster and eliminates the numerical error that would accumulate if the bases specified by the tall $\widetilde{\mathbf{Q}}$ matrix were rotated on each update.

With the new expression of the decomposition form in Eq. (III.11), we update the process of our ILRED algorithm.

1) When the norm of the residual vector is small: If the norm of the residual vector is small, i.e., $p = \|\mathbf{e}\| \le \varepsilon$, based on the preceding discussion in Section. III-A, it is imperative to ensure that the product $\mathbf{Q} = \widetilde{\mathbf{Q}}\widehat{\mathbf{Q}}$ remains orthogonal. Referring to equations (III.6) and (III.7), we can deduce that the right-side update must adhere to the following form

$$\widetilde{\mathbf{Q}}_{\text{new}}\,\widehat{\mathbf{Q}}_{\text{new}} = \begin{bmatrix} \widetilde{\mathbf{Q}}_{\text{old}}\,\widehat{\mathbf{Q}}_{\text{old}} & 0 \\ 0 & 1 \end{bmatrix}\mathbf{X}, \tag{III.12}$$

where $\widetilde{\mathbf{Q}}_{\text{old}}$ and $\widehat{\mathbf{Q}}_{\text{old}}$ correspond to the original matrix $\mathbf{M}_{\text{old}}$, and $\widetilde{\mathbf{Q}}_{\text{new}}$ and $\widehat{\mathbf{Q}}_{\text{new}}$ correspond to the updated matrix $\mathbf{M}_{\text{new}}$. $\mathbf{X}$ is a variable matrix that takes different entry values in different cases. When the smallest eigenvalue of $\mathbf{Y}_1$ is small, as in the scenario of Eq. (III.6), $\mathbf{X} = \mathbf{Q}_1(:, 1:k)$. In this case, the rank of the updated matrix does not increase. When the smallest eigenvalue of $\mathbf{Y}_1$ is significant and cannot be ignored, we set $\mathbf{X} = \mathbf{Q}_1$, corresponding to the scenario of Eq. (III.7). In this case, the rank of the updated matrix increases.

We can reduce the computational complexity of Eq. (III.12) by keeping on updating a small pseudo-inverse matrix $\widehat{\mathbf{Q}}^{+}_{\text{old}}$. When the rank does not increase, we can further reduce the complexity by splitting $\mathbf{Q}_1(:, 1:k) \in \mathbb{R}^{(k+1)\times k}$ into the form

$$\mathbf{Q}_1(:, 1:k) = \begin{bmatrix} \mathbf{W} \in \mathbb{R}^{k\times k} \\ \mathbf{w} \in \mathbb{R}^{1\times k} \end{bmatrix}, \tag{III.13}$$

where submatrix $\mathbf{W}$ is a linear transform that will be applied to $\widehat{\mathbf{Q}}_{\text{old}}$, and row-vector $\mathbf{w}$ is the subspace projection of the new data vector. By substituting Eq. (III.13) into Eq. (III.12), we have

$$\begin{aligned} \widetilde{\mathbf{Q}}_{\text{new}}\,\widehat{\mathbf{Q}}_{\text{new}} &= \begin{bmatrix} \widetilde{\mathbf{Q}}_{\text{old}}\,\widehat{\mathbf{Q}}_{\text{old}} & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \mathbf{W} \\ \mathbf{w} \end{bmatrix} \\ &= \begin{bmatrix} \widetilde{\mathbf{Q}}_{\text{old}}\,\widehat{\mathbf{Q}}_{\text{old}}\,\mathbf{W} \\ \mathbf{w} \end{bmatrix} \\ &= \begin{bmatrix} \widetilde{\mathbf{Q}}_{\text{old}} \\ \mathbf{w}\mathbf{W}^{+}\widehat{\mathbf{Q}}^{+}_{\text{old}} \end{bmatrix}\widehat{\mathbf{Q}}_{\text{old}}\,\mathbf{W}. \end{aligned} \tag{III.14}$$

Noting sizes of matrices $\widehat{\mathbf{Q}}_{\text{old}}\,\mathbf{W} \in \mathbb{R}^{k\times k}$ and matrix $\begin{bmatrix} \widetilde{\mathbf{Q}}_{\text{old}} \\ \mathbf{w}\mathbf{W}^{+}\widehat{\mathbf{Q}}^{+}_{\text{old}} \end{bmatrix} \in \mathbb{R}^{(n+1)\times k}$, we let $\widehat{\mathbf{Q}}_{\text{new}} = \widehat{\mathbf{Q}}_{\text{old}}\,\mathbf{W}$ and $\widetilde{\mathbf{Q}}_{\text{new}} = \begin{bmatrix} \widetilde{\mathbf{Q}}_{\text{old}} \\ \mathbf{w}\widehat{\mathbf{Q}}^{+}_{\text{new}} \end{bmatrix}$. Then the resulting right-side update becomes

$$\begin{aligned} \widehat{\mathbf{Q}}_{\text{new}} &\leftarrow \widehat{\mathbf{Q}}_{\text{old}}\,\mathbf{W}; \\ \widehat{\mathbf{Q}}^{+}_{\text{new}} &\leftarrow \mathbf{W}^{+}\widehat{\mathbf{Q}}^{+}_{\text{old}}\,; \\ \widetilde{\mathbf{Q}}_{\text{new}} &\leftarrow \begin{bmatrix} \widetilde{\mathbf{Q}}_{\text{old}} \\ \mathbf{w}\widehat{\mathbf{Q}}^{+}_{\text{new}} \end{bmatrix} \\ \boldsymbol{\Sigma} &\leftarrow \boldsymbol{\Sigma}_1(1:k, 1:k). \end{aligned} \tag{III.15}$$

Conveniently, the pseudo-inverse $\mathbf{W}^{+}$ can be computed in $\mathcal{O}\left(k^2\right)$-time using only matrix-vector and vector-vector products via the identity

$$\mathbf{W}^{+} = \mathbf{W}^{\top} + \frac{\mathbf{w}^{\top}}{1 - \|\mathbf{w}\|^2}\left(\mathbf{w}\mathbf{W}^{\top}\right). \tag{III.16}$$

When the update is rank-increasing, Eq. (III.12) can be decomposed as

$$\begin{aligned} \widetilde{\mathbf{Q}}_{\text{new}}\,\widehat{\mathbf{Q}}_{\text{new}} &= \begin{bmatrix} \widetilde{\mathbf{Q}}_{\text{old}}\,\widehat{\mathbf{Q}}_{\text{old}} & 0 \\ 0 & 1 \end{bmatrix}\mathbf{Q}_1 \\ &= \begin{bmatrix} \widetilde{\mathbf{Q}}_{\text{old}} & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \widehat{\mathbf{Q}}_{\text{old}} & 0 \\ 0 & 1 \end{bmatrix}\mathbf{Q}_1 \end{aligned} \tag{III.17}$$

The right-side update in Eq. (III.12) can be replaced by

$$\begin{aligned} \widehat{\mathbf{Q}}_{\text{new}} &\leftarrow \begin{bmatrix} \widehat{\mathbf{Q}}_{\text{old}} & 0 \\ 0 & 1 \end{bmatrix}\mathbf{Q}_1; \\ \widehat{\mathbf{Q}}^{+}_{\text{new}} &\leftarrow \mathbf{Q}_1^{\top}\begin{bmatrix} \widehat{\mathbf{Q}}^{+}_{\text{old}} & 0 \\ 0 & 1 \end{bmatrix}; \\ \widetilde{\mathbf{Q}}_{\text{new}} &\leftarrow \begin{bmatrix} \widetilde{\mathbf{Q}}_{\text{old}} & 0 \\ 0 & 1 \end{bmatrix}. \end{aligned} \tag{III.18}$$

2) When the norm of the residual vector is large: If the norm of the residual vector is large, i.e., $p = \|\mathbf{e}\| > \varepsilon$. Referring to equations (III.9) and (III.10), the update must satisfy

$$\widetilde{\mathbf{Q}}_{\text{new}}\,\widehat{\mathbf{Q}}_{\text{new}} = \begin{bmatrix} \widetilde{\mathbf{Q}}_{\text{old}}\,\widehat{\mathbf{Q}}_{\text{old}} & \widetilde{\mathbf{e}} & 0 \\ 0 & 0 & 1 \end{bmatrix}\mathbf{X}, \tag{III.19}$$

where $\mathbf{X}$ is a variable matrix that takes different values in different cases. When the rank remains unchanged, set $\mathbf{X} = \mathbf{Q}_2(:, 1:k)$ and split $\mathbf{X} = \begin{bmatrix} \mathbf{W} \in \mathbb{R}^{(k+1)\times k} \\ \mathbf{w} \in \mathbb{R}^{1\times k} \end{bmatrix}$. In cases

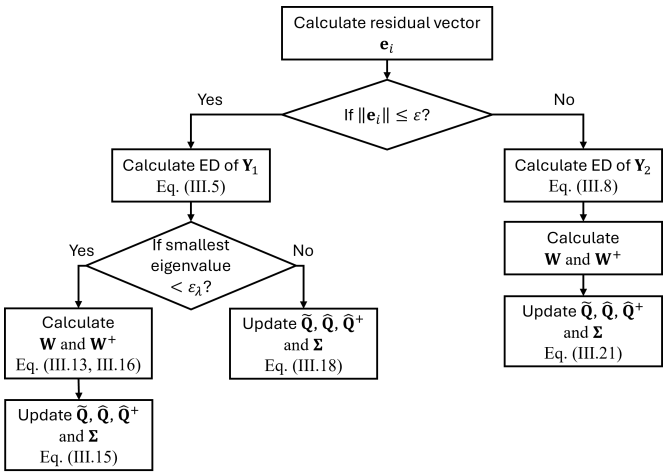

Fig. 4. Flow diagram of one iteration in algorithm 2.

where the rank increases, we let $\mathbf{X} = \mathbf{Q}_2(:, 1:k+1)$ and split it into $\mathbf{X} = \begin{bmatrix} \mathbf{W} \in \mathbb{R}^{(k+1)\times(k+1)} \\ \mathbf{w} \in \mathbb{R}^{1\times(k+1)} \end{bmatrix}$. Now we have

$$\begin{aligned}
\widetilde{\mathbf{Q}}_{\text{new}}\,\widehat{\mathbf{Q}}_{\text{new}} &= \begin{bmatrix} \widetilde{\mathbf{Q}}_{\text{old}}\,\widehat{\mathbf{Q}}_{\text{old}} & \widetilde{\mathbf{e}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{W} \\ \mathbf{w} \end{bmatrix} \\
&= \begin{bmatrix} \begin{bmatrix} \widetilde{\mathbf{Q}}_{\text{old}}\widehat{\mathbf{Q}}_{\text{old}} & \widetilde{\mathbf{e}} \end{bmatrix} \mathbf{W} \\ \mathbf{w} \end{bmatrix} \\
&= \begin{bmatrix} \begin{bmatrix} \widetilde{\mathbf{Q}}_{\text{old}} & \widetilde{\mathbf{e}} \end{bmatrix} \begin{bmatrix} \widehat{\mathbf{Q}}_{\text{old}} & 0 \\ 0 & 1 \end{bmatrix} \mathbf{W} \\ \mathbf{w} \end{bmatrix} \\
&= \begin{bmatrix} \begin{bmatrix} \widetilde{\mathbf{Q}}_{\text{old}} & \widetilde{\mathbf{e}} \end{bmatrix} \\ \mathbf{w}\mathbf{W}^+ \begin{bmatrix} \widehat{\mathbf{Q}}^+_{\text{old}} & 0 \\ 0 & 1 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \widehat{\mathbf{Q}}_{\text{old}} & 0 \\ 0 & 1 \end{bmatrix} \mathbf{W}
\end{aligned} \tag{III.20}$$

The update is simply

$$\begin{aligned}
\widehat{\mathbf{Q}}_{\text{new}} &\leftarrow \begin{bmatrix} \widehat{\mathbf{Q}}_{\text{old}} & 0 \\ 0 & 1 \end{bmatrix} \mathbf{W}; \\
\widehat{\mathbf{Q}}^+_{\text{new}} &\leftarrow \mathbf{W}^+ \begin{bmatrix} \widehat{\mathbf{Q}}^+_{\text{old}} & 0 \\ 0 & 1 \end{bmatrix}; \\
\widetilde{\mathbf{Q}}_{\text{new}} &\leftarrow \begin{bmatrix} \begin{bmatrix} \widetilde{\mathbf{Q}}_{\text{old}} & \widetilde{\mathbf{e}} \end{bmatrix} \\ \mathbf{w}\widehat{\mathbf{Q}}^+_{\text{new}} \end{bmatrix}.
\end{aligned} \tag{III.21}$$

The flow diagram of one update iterations is shown in Fig. 4.

For the case of multiple dimension increase, we will update the ED result by multiple iterations using the introduced algorithm. Assuming we have the ED matrices $\mathbf{Q}_{\text{init}} \in \mathbb{R}^{n\times k}$ and $\mathbf{\Sigma}_{\text{init}} \in \mathbb{R}^{k\times k}$ for the rank k initial matrix $\mathbf{M}_{\text{init}} \in \mathbb{R}^{n\times n}$, the final matrix $\mathbf{M}_{\text{final}} \in \mathbb{R}^{(n+\ell)\times(n+\ell)}$, where $\ell \ge 1$, we summarized our proposed fast ILRED algorithm for multiple dimension increase case in Alg. 2.

IV. Analysis

In this section, we perform an error analysis and complexity analysis of the incremental algorithms.

Algorithm 2 Fast Incremental Eigen-Decomposition Algorithm (ILRED-FAST)

Input: Decomposition matrices $\mathbf{Q}_{\text{init}}$, diagonal eigenvalue matrix $\mathbf{\Sigma}_{\text{init}}$, additional vectors $\{\boldsymbol{\alpha}_i, i \in 1, \cdots, \ell\}$, additional singular values $\{d_i, i \in 1, \cdots, \ell\}$, tolerance ε for residual vector and threshold ε_λ for eigenvalue;
1. Set $\widetilde{\mathbf{Q}}$ and $\mathbf{\Sigma}$ as $\mathbf{Q}_{\text{init}}$ and $\mathbf{\Sigma}_{\text{init}}$ respectively, and set both $\widehat{\mathbf{Q}}$ and $\widehat{\mathbf{Q}}^+$ as identity matrix $\mathbf{I}$;
for i from 1 to ℓ **do**
  2. Calculate the residual vector $\mathbf{e}_i$ and its norm using Eq. (III.3);
  if $p = \|\mathbf{e}\| \le \varepsilon$ **then**
    3. Calculate the ED of $\mathbf{Y}_1$ defined in Eq. (III.5);
    if the smallest eigenvalue of $\mathbf{Y}_1$ is smaller than ε_λ **then**
      4. Calculate $\mathbf{W}$ and $\mathbf{W}^+$ based on $\mathbf{Q}_1$ using Eq. (III.13) and Eq. (III.16);
      5. Update $\widetilde{\mathbf{Q}}$, $\widehat{\mathbf{Q}}$, $\widehat{\mathbf{Q}}^+$ and $\mathbf{\Sigma}$ using Eq. (III.15);
    else
      6. Update $\widetilde{\mathbf{Q}}$, $\widehat{\mathbf{Q}}$, $\widehat{\mathbf{Q}}^+$ and $\mathbf{\Sigma}$ using Eq. (III.18);
    end if
  else
    6. Calculate the ED of $\mathbf{Y}_2$ defined in Eq. (III.8);
    7. Calculate $\mathbf{W}$ and $\mathbf{W}^+$ based on $\mathbf{Q}_2$;
    8. Update $\widetilde{\mathbf{Q}}$, $\widehat{\mathbf{Q}}$, $\widehat{\mathbf{Q}}^+$ and $\mathbf{\Sigma}$ using Eq. (III.21);
  end if
end for
9. Calculate $\mathbf{Q}$ using $\widetilde{\mathbf{Q}}$ and $\widehat{\mathbf{Q}}$;
Output: Decomposition matrix $\mathbf{Q}$ and diagonal eigenvalue matrix $\mathbf{\Sigma}$ as the approximated ED of $\mathbf{M}_{\text{final}}$.

A. Error Analysis

In a typical application of the eigen-updating algorithm, many new columns and rows of data are added so that the incremental ED is updated many times. We assume that we are at one step of this procedure where the rank of the initial matrix $\mathbf{M}_{\text{old}}$ is k, and we have an existing error bound. We prove that Alg. 1 produces a correct update of the error bound.

Theorem 2. Let $\mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top$ represent the eigen approximation of matrix $\mathbf{M}_{\text{old}}$. Define p as $\|\boldsymbol{\alpha} - \mathbf{Q}\mathbf{Q}^\top\boldsymbol{\alpha}\|$, and let $\mathbf{Q}_{\text{new}}\mathbf{\Sigma}_{\text{new}}\mathbf{Q}^\top_{\text{new}}$ be the eigen approximation of the matrix $\mathbf{M}_{\text{new}}$, where $\mathbf{M}_{\text{new}}$ is given by $\begin{bmatrix} \mathbf{M}_{\text{old}} & \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^\top & d \end{bmatrix}$, using the incremental Eigen-decomposition (ILRED) algorithm. Then, the Frobenius norm of the difference between $\mathbf{M}_{\text{new}}$ and $\mathbf{Q}_{\text{new}}\mathbf{\Sigma}_{\text{new}}\mathbf{Q}^\top_{\text{new}}$ is given by:

$$\mathbf{\Delta}_{\mathbf{M}_{\text{new}}} \le \begin{cases} \mathbf{\Delta}_{\mathbf{M}_{\text{old}}} + \sqrt{2}p, & \text{if } p \le \varepsilon \text{ and NET}, \\ \mathbf{\Delta}_{\mathbf{M}_{\text{old}}} + \sqrt{2}p + \lambda_1, & \text{if } p \le \varepsilon \text{ and ET}, \\ \mathbf{\Delta}_{\mathbf{M}_{\text{old}}}, & \text{if } p > \varepsilon \text{ and NET}, \\ \mathbf{\Delta}_{\mathbf{M}_{\text{old}}} + \lambda_2, & \text{if } p > \varepsilon \text{ and ET}, \end{cases}$$

where $\mathbf{\Delta}_{\mathbf{M}_{\text{old}}} = \left\|\mathbf{M}_{\text{old}} - \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top\right\|_F$ represents the differences between matrix $\mathbf{M}_{\text{old}}$ and its eigen approximation, and $\mathbf{\Delta}_{\mathbf{M}_{\text{new}}} = \left\|\mathbf{M}_{\text{new}} - \mathbf{Q}_{\text{new}}\mathbf{\Sigma}_{\text{new}}\mathbf{Q}^\top_{\text{new}}\right\|_F$ is the differences between matrix $\mathbf{M}_{\text{new}}$ and its eigen approximation. ET stands

for 'eigenvalue truncation is applied' and NET stands for 'no eigenvalue truncation is applied'.

Proof. Please see Appendix B for the proof. □

The result above explains the update of the error bound in one step of Alg. 1. Now we assume the ILRED algorithm is initialized exactly when $k = 1$, and then the algorithm is applied for a sequence of added columns and rows, the final estimation error is given as follows.

Theorem 3. Let the thresholds ε and ε_λ be fixed positive constants. If T_p represents the total number of times p-truncation is applied and T_λ represents the total number of times the eigenvalue truncation is applied, then

$$\left\|\mathbf{M}_{\text{old}} - \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top\right\|_F \leq \sqrt{2}T_p\varepsilon + T_\lambda\varepsilon_\lambda$$

Proof. Please see Appendix C for the proof. □

B. Complexity Analysis

In this subsection, we conduct a detailed analysis of the computational complexity associated with our proposed algorithms. The evaluation of computational complexity is measured in terms of the order of multiplication operations required. This metric provides a basis for assessing the efficiency and scalability of our proposed algorithms. In the following analysis, we assume the rank of the initial matrix $\mathbf{M}_{\text{old}} \in \mathbb{R}^{n\times n}$ is k, and the size of the final matrix $\mathbf{M}_{\text{final}}$ is $(n+\ell)$-by-$(n+\ell)$, i.e., we have ℓ iterations in our algorithms. We start by analyzing the first iteration of our proposed algorithms.

1) Complexity Analysis of the ILRED-BASIC Algorithm: First, the complexity order for calculating the residual vector $\mathbf{e}_i$ defined in Eq. (III.3) is $O(nk)$, since the complexity of the product between matrix $\mathbf{Q}^\top \in \mathbb{R}^{k\times n}$ and vector $\boldsymbol{\alpha} \in \mathbb{R}^{n\times 1}$ is $O(nk)$, and the complexity of the product between $\mathbf{Q} \in \mathbb{R}^{n\times k}$ and $(\mathbf{Q}^\top\boldsymbol{\alpha}) \in \mathbb{R}^{k\times 1}$ is also $O(nk)$. The complexity of calculating the norm of $\mathbf{e}_i \in \mathbb{R}^{n\times 1}$ is $O(n)$.

Then, the complexity order of calculating the ED of either $\mathbf{Y}_1 \in \mathbb{R}^{(k+1)\times(k+1)}$ and $\mathbf{Y}_2 \in \mathbb{R}^{(k+2)\times(k+2)}$ is $O(k^3)$. Finally, the complexity order of updating $\mathbf{Q}$ and $\mathbf{\Sigma}$ is $O(nk^2)$, with the worst-case scenario outlined by (III.10). This scenario has the multiplication between matrix

$$\begin{bmatrix} \mathbf{Q} & \widetilde{\mathbf{e}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{(n+1)\times(k+2)}$$

and matrix $\mathbf{Q}_2(:, 1:k+1) \in \mathbb{R}^{(k+2)\times(k+1)}$. Therefore, the complexity order for the first iteration of the ILRED-BASIC algorithm is $O(nk^2+k^3)$. Because we have ℓ iterations in our algorithm, the overall complexity order for the ILRED-BASIC algorithm is given as follows.

Theorem 4. Let $\mathbf{M}_{\text{old}} \in \mathbb{R}^{n\times n}$ be the initial matrix with rank k and $\mathbf{M}_{\text{final}} \in \mathbb{R}^{(n+\ell)\times(n+\ell)}$ be the final matrix, then the overall complexity order of ILRED-BASIC algorithm is $O(k\ell^3 + k^2\ell^2 + k^3\ell + kn\ell^2 + k^2n\ell + n\ell^3 + \ell^4)$

Proof. Please see Appendix D for the proof. □

By this theorem, when the number of iterations ℓ is relatively small, the complexity order of the ILRED-BASIC algorithm is dominated by the terms $(nk^2 + k^3)\ell$, which is increased approximated linearly with the number of iterations. When ℓ is close to the value of n, the complexity order is dominated by the terms $(k+n)\ell^3 + \ell^4$, which increases fast.

2) Complexity Analysis of the ILRED-FAST Algorithm: Similarly, the complexity order for calculating the residual vector $\mathbf{e}_i$ is $O(nk)$, and the complexity of calculating the norm of $\mathbf{e}_i$ is $O(n)$. Then, the complexity order of calculating the ED of either $\mathbf{Y}_1 \in \mathbb{R}^{(k+1)\times(k+1)}$ and $\mathbf{Y}_2 \in \mathbb{R}^{(k+2)\times(k+2)}$ is $O(k^3)$. Next, the matrix $\mathbf{W}$ is taken by splitting either $\mathbf{Q}_1$ or $\mathbf{Q}_2$ and it take the complexity of $O(1)$. The complexity of getting pseudo-inverse $\mathbf{W}^+$ given by (III.16) is $O(k^2)$, because the complexity of the product between vector $\mathbf{w} \in \mathbb{R}^{1\times k}$ and matrix $\mathbf{W}^\top$ is $O(k^2)$, and the complexity of the product between the vector $\frac{\mathbf{w}^\top}{1-\|\mathbf{w}\|^2}$ and vector $(\mathbf{w}\mathbf{W}^\top)$ is also $O(k^2)$.

Finally, the complexity of updating $\widetilde{\mathbf{Q}}$, $\widehat{\mathbf{Q}}$, $\widehat{\mathbf{Q}}^+$ and $\mathbf{\Sigma}$ is $O(k^3)$, with the worst scenario is given by (III.21). The update of either $\widehat{\mathbf{Q}}_{\text{new}}$ and $\widehat{\mathbf{Q}}^+_{\text{new}}$ has the complexity of $O(k^3)$ with the product between two matrices, while the update of $\widetilde{\mathbf{Q}}$ has the complexity of $O(k^2)$ with the product between a vector and a matrix. In summary, the complexity order for the first iteration of the ILRED-FAST algorithm is $O(nk+k^3)$, considering that the value of k^2 may be close to the value of n. Since we have ℓ iterations in our algorithm, the overall complexity order for the ILRED-FAST algorithm is given as follows.

Theorem 5. Let $\mathbf{M}_{\text{old}} \in \mathbb{R}^{n\times n}$ be the initial matrix with rank k and $\mathbf{M}_{\text{final}} \in \mathbb{R}^{(n+\ell)\times(n+\ell)}$ be the final matrix, then the overall complexity order of ILRED-FAST algorithm is $O(k\ell^3 + k^2\ell^2 + k^3\ell + \ell^4 + n\ell^2 + kn\ell)$

Proof. Please see Appendix E for the proof. □

When the number of iterations ℓ is relatively small, the complexity order of the ILRED-FAST algorithm is dominated by the terms $(kn + k^3)\ell$, which is increased approximated linearly with the number of iterations and smaller than ILRED-BASIC. When ℓ is close to the value of n, the complexity order is dominated by the terms $k\ell^3 + \ell^4$, which is also smaller than ILRED-BASIC algorithm.

V. Experiments

We now present the experimental results of the proposed fast ILRED algorithm, comparing its performance in both synthetic and real datasets against existing ED approximation approaches. Specifically, we evaluate our ILRED-FAST algorithm alongside other state-of-the-art methods, including efficient eigen-updating (EEU) algorithm presented in [15], the TRIP-BASIC and TRIP algorithm outlined in [17], as well as the conventional eigen-decomposition (EIG) algorithm.

A. Synthetic Dataset

In this part, we evaluate the error performance and runtime of our proposed ILRED algorithms using positive definite symmetric matrices. The reason for selecting such matrices is to demonstrate the algorithm's versatility across different matrix sizes. We generate these random symmetric matrices as follows: we first generate an n-by-n size random matrix whose

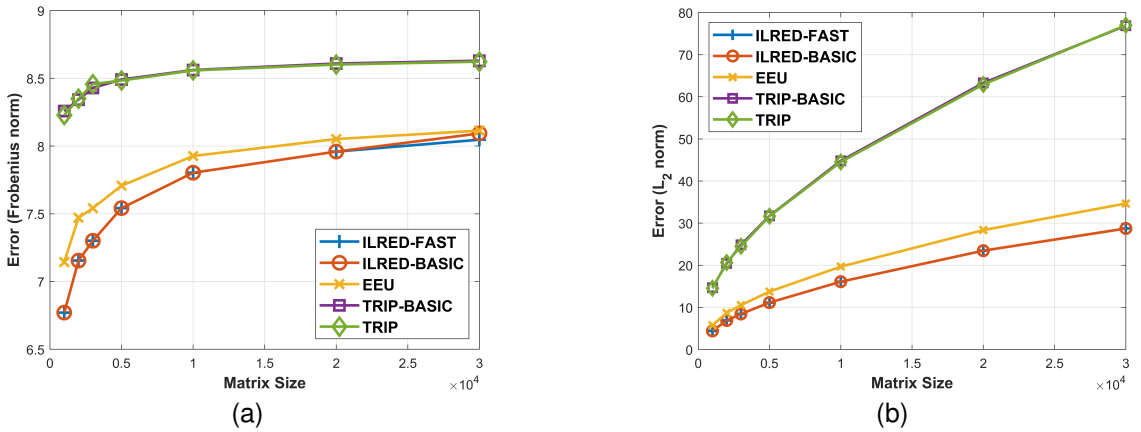

Fig. 5. Estimation errors of the ED algorithm on synthetic dataset. (a) Average estimation errors of eigenvector matrix $\mathbf{Q}$, measured by Frobenius norm. (b) Average estimation errors of eigenvalue matrix $\mathbf{\Sigma}$, measured by L_2 norm.

elements are randomly sampled from a uniform distribution from 0 to 1. Sequentially, we calculate the mean of this random matrix and its transpose to form a symmetric matrix. We then calculate the ED matrices $\mathbf{Q}$ and $\mathbf{\Sigma}$ of the symmetric matrix using conventional EIG algorithm. Next, we set the elements of $\mathbf{\Sigma}$ to its absolute value. Finally, we use $\mathbf{Q}$ and updated $\mathbf{\Sigma}$ to generate a positive definite symmetric matrix.

Throughout our experiments, we set the final matrix of all ED algorithms to an n-by-n size, while the initial matrix $\mathbf{M}_{\text{init}}$ is configured as the top left submatrix of size $0.9n$-by-$0.9n$. We focus on updating top K eigen-pairs to match the task in Section V-B. The focus is to truncate the input decomposition matrices $\mathbf{Q}_{\text{init}}$ and $\mathbf{\Sigma}_{\text{init}}$ by keeping the top K eigen-pairs, which correspond to the eigenvalues of the initial matrix $\mathbf{M}_{\text{init}}$ with the largest magnitudes. After getting the estimation result of the ED algorithm, we truncate the output matrices $\mathbf{Q}$ and $\mathbf{\Sigma}$ in the same way and compare them with the ground truth result given by EIG algorithm on the final matrix. To reduce the impact of the randomness, we conduct 100 Monte Carlo runs for each matrix size and use the average estimation error as the metric. To measure the estimation errors on eigenvector matrix $\mathbf{Q}$, we first reorder the columns in $\mathbf{Q}$ to find the best match with the ground truth. We then calculate the Frobenius norm of the difference between the reordered matrix and the ground truth. In this way, we can avoid the order mismatch of the eigenvectors caused by the small turbulence on the estimation of eigenvalues. On the other hand, we use the L_2 norm to measure the difference of eigenvalues with the ground truth. The average estimation errors of all ED algorithms are shown in Fig. 5. Our proposed ILRED-BASIC and ILRED-FAST similarly have the lowest estimation errors on all matrix sizes ranging from 1,000 to 30,000, which demonstrates the accuracy of our proposed algorithms. Considering the fact that the main difference between the EEU algorithm and our proposed ILRED-FAST algorithm is the error correction branch, this performance improvement demonstrates the effect of the error correction branch in our proposed algorithm. Also, the TRIP and the TRIP-BASIC algorithms perform badly compared with other algorithms, showing that these two algorithms are not suitable for ED updating problems when the size of the matrix is increasing.

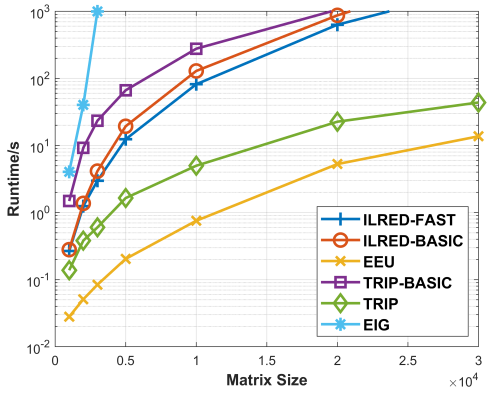

Fig. 6. Average runtime of the ED algorithm on synthetic dataset.

We also collect the average runtime of all ED algorithms for different matrix sizes. All the algorithms are implemented in MATLAB and executed on the same computer equipped with i5-9600K CPU at 3.7 GHz. As shown in Fig. 6, all ED algorithms are much faster than the conventional EIG algorithm, which re-computes the eigen-decomposition each time the matrix size is increased by one. The runtime of ILRED-BASIC algorithm is larger than that of the ILRED-FAST algorithm, which is consistent with the findings from the complexity analysis presented in Section IV-B. The runtime of both of our proposed ILRED algorithms are smaller than TRIP-BASIC, but larger than TRIP and EEU algorithms. Such higher runtime of our proposed algorithms is partially caused by the high number of loops in the implementation. Considering both TRIP and EEU are updating the matrix changes in one step, they have a lower number of loops than our proposed ILRED-FAST algorithm.

B. Real Datasets

In this subsection, we assess the error performance and runtime of our proposed ILRED algorithms in real multimedia datasets. Specifically, we subject the ED algorithms on dynamic point clouds, wireframe datasets, and hyperspectral

images as examples for graph-based applications, including spectral clustering and low-pass filtering. Despite the adjacency matrices for graphs built on real datasets are typically high rank, our focus lies on updating top K eigen-pairs, considering the fact that the top K eigen-pairs are commonly used in spectral clustering and low-pass graph filtering algorithms. For other GSP applications using Laplacian matrices instead of adjacency matrices, we can take the pseudoinverse of the Laplacian matrix [33], [34] to convert the smallest eigenvalues to the largest ones before applying our algorithm on the top K eigenpairs. To ensure a fair comparison, we standardize the comparison process by truncating the output matrices $\mathbf{Q}$ and $\mathbf{\Sigma}$ of our proposed ILRED-FAST algorithm by selecting the first K column vectors and diagonal elements, respectively.

In our experiments on all three datasets, we begin by constructing the adjacency matrix based on the attribute similarities, using it as the final matrix for all ED algorithms. Subsequently, we randomly drop some nodes of the data, resulting in a truncated adjacency matrix with the corresponding rows and columns removed. We then compute the top K eigen-pairs of this truncated adjacency matrix using conventional EIG algorithm as the input of all ED algorithms. Finally, we use the output of these ED algorithms for spectral clustering and low-pass filtering to assess their error performance.

For experiments on spectral clustering, we employ the approximated ED results $\mathbf{Q}$ to cluster the data, comparing the results with the cluster outcomes using the conventional EIG algorithm. The number of clustering errors serves as the metric to assess error performance. Since spectral clustering algorithm only uses the estimated eigenvectors, these results reflect the accuracy of the estimated eigenvectors.

In low-pass filtering experiments, we use both ideal graph-based filter and Haar-like graph-based filter based on the approximated top K eigen-pairs of ED algorithms to process the graph signal based on the vector of the first attribute on all nodes. The ideal filter uses all K estimated eigenvectors, while the Haar-like filter also uses the corresponding estimated eigenvalues. The ground truth for the filtered spatial domain signal is generated using eigen-pairs from the conventional EIG algorithm, and the L_2 norm of the difference between the filtered spatial domain signal using ED algorithms and ground truth as the metrics to assess error performance. The results of ideal graph-based filters reflect the accuracy of estimating eigenvectors of ED algorithms, while the results of Haar-like graph-based filters reflect the accuracy of estimating both eigenvectors and the exact value of all eigenvalues.

1) Dynamic Point Clouds: Dynamic point clouds have various applications in autonomous driving [24], virtual reality [25] and medical imaging [26]. An example of dynamic point cloud with additional nodes in successive frames is shown in Fig. 7. In this example, the points on the right foot are not captured by the sensors in the first frame. These additional points in the second frame result in the addition of nodes and edges in the new graph, which corresponds to the additional rows and columns in the new adjacency matrix. As shown in the Fig. 7(b), dynamic graph is a natural method to capture the underlying relationship of points in dynamic point cloud. As a result, graph-based analyzing and processing methods, such as spectral clustering and graph-based filtering, are popular ways of processing dynamic point clouds. The authors in [35] utilize spectral clustering to separate the motion flows in their proposed object detection and extraction framework on mobile lidar data. A robust dynamic point cloud segmentation routine is proposed in [36], where the spectral clustering is used to generate the initial segmentation. A graph-based low-pass filtering method is proposed in [37] for point cloud denoising.

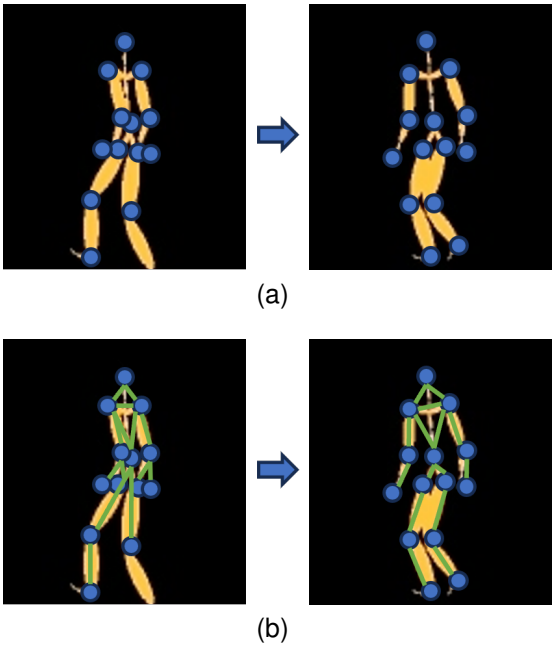

Fig. 7. Example of dynamic point cloud with additional nodes in successive frames. (a) Example of dynamic point cloud, the points on the right foot are not captured by the sensors in the first frame. (b) Example of the graphs built on the dynamic point cloud, the additional points in the second frame resulting in the addition of nodes and edges in the new graph, corresponding to the additional rows and columns in the new adjacency matrix.

In this paper, we only focus on the cases when the size of the point cloud increases. Therefore, we use the static point clouds in the Mythological Creatures database [27], [28] to simulate a dynamic point cloud with the addition of nodes in successive frames. We conduct 10 Monte Carlo runs on each point cloud. In each round we first build the ϵ-neighborhood graph on the point cloud with the radius of neighborhood ϵ as three times of the intrinsic distance d_r as Eq. (II.1). The distance between two points is measured by the L_2 norm of the difference on attributes of these two points. Then we calculate the adjacency matrix of the constructed graph using the Gaussian kernel. This adjacency matrix is the final matrix of all ED algorithms. Next, we randomly drop 1% of the nodes in the point cloud, and remove its corresponding rows and columns in the adjacency matrix. We then calculate the top K eigen-pairs of this truncated adjacency matrix $\mathbf{A}_{\text{init}}$ to generate the input decomposition matrix $\mathbf{Q}_{\text{init}}$ and $\mathbf{\Sigma}_{\text{init}}$. In our test the number of tracking eigen-pairs K is set to 100. We summarize this ED updating algorithm in Alg. 3.

After getting the approximated ED results $\mathbf{Q}$ and $\mathbf{\Sigma}$ of the final matrix, we use them to do spectral clustering, Haar-like low-pass filtering and ideal low-pass filtering compare them with the ground truth results using the top K eigen-pairs of the

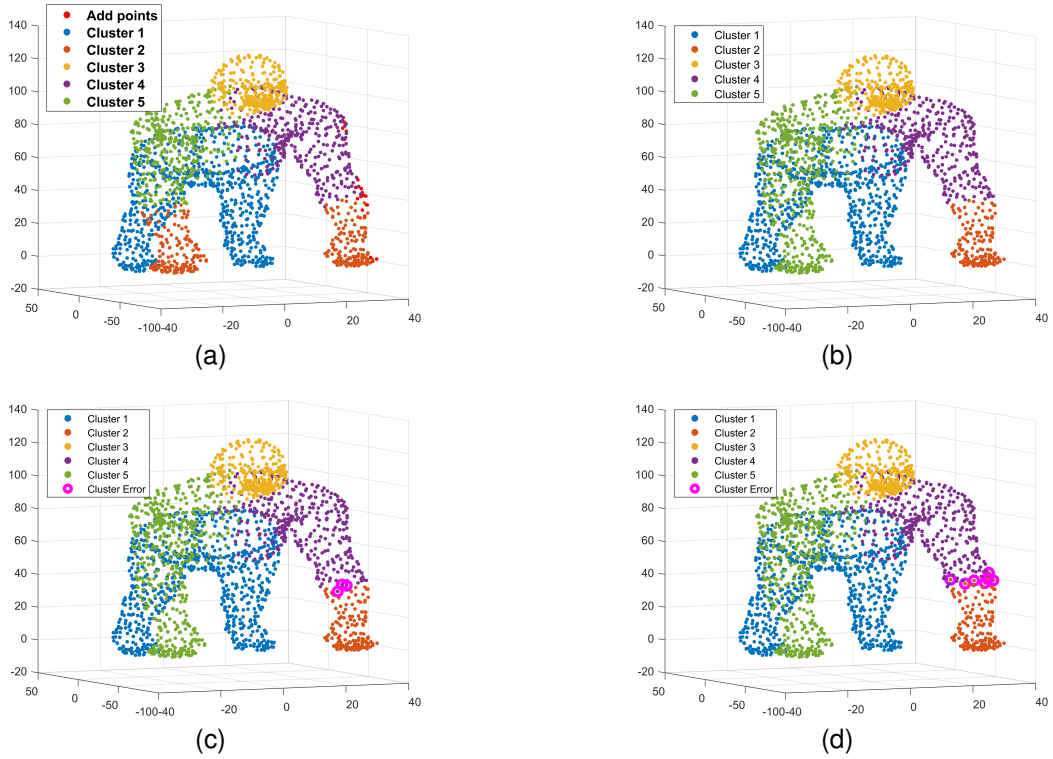

Fig. 8. Example of spectral clustering result in our experiments. (a) Clustering result of the original point cloud using the ground truth ED result, updated points are labeled in red. (b) Clustering result of the updated point cloud using ground truth. (c) Clustering result of the updated point cloud using estimated ED results from ILRED-FAST algorithm, clustering errors are marked in pink. (d) Clustering result of the updated point cloud using estimated ED results from ILRED-BASIC algorithm, clustering errors are marked in pink.

Algorithm 3 Fast Incremental Eigen-Decomposition Updating Algorithm for Dynamic Point Cloud

Input: Coordinates of the updated points $\mathbf{P}_u$, adjacency matrix of the point cloud at the current frame $\mathbf{A}_{\text{init}}$, number of tracking eigen-pairs K, intrinsic resolution d_r;
Output: Decomposition matrix $\mathbf{Q}$ and diagonal eigenvalue matrix $\mathbf{\Sigma}$ as the approximated ED of the adjacency matrix of the updated matrix $\mathbf{A}_{\text{final}}$;
1. For each updated point whose coordinates are in $\mathbf{P}_u$, calculate the additional vector $\boldsymbol{\alpha}_i$ as Eq. (II.1), set all singular values $\{d_i\}$ as 1;
2. Calculate $\mathbf{Q}_{\text{init}}$ and $\mathbf{\Sigma}_{\text{init}}$ containing the top K eigen-pairs of $\mathbf{A}_{\text{init}}$ using conventional EIG algorithm;
3. Use Alg. 1 or Alg. 2 to calculate the approximated ED results $\mathbf{Q}$ and $\mathbf{\Sigma}$ for the updated frame;
4. Trim $\mathbf{Q}$ and $\mathbf{\Sigma}$ by keeping the top K eigen-pairs.

full-sized adjacency matrix. For spectral clustering results, the error performance is measured by the number of points that are incorrectly clustered. For low-pass filtering results, the error performance is measured by the L_2 norm between the filtered signals using approximated results and ground truth.

The average result is shown in Table. II. Our proposed ILRED-BASIC algorithm has the lowest error in L_2 norms, while the ILRED-FAST algorithm has the lowest error in terms of clustering errors, which demonstrates the accuracy of our proposed algorithms on both eigenvalues and eigenvectors

TABLE II
AVERAGE CLUSTERING ERRORS AND LOW-PASS FILTER ERRORS ON DYNAMIC POINT CLOUDS

	ILRED-FAST	ILRED-BASIC	EEU	TRIP-BASIC	TRIP
Number of Clustering Errors	**31.0270**	31.0676	41.3378	101.7568	101.2838
L2 (Haar)	51.6431	**51.6416**	60.4235	111.5380	111.2226
L2 (Ideal)	68.1637	**68.1631**	84.0457	187.3133	186.9413

in dynamic point clouds. ILRED-FAST also has a superior performance over the existing approaches. Compared with the EEU algorithm, our proposed ILRED algorithms have significant performance improvement. Considering the fact that the main difference between the EEU algorithm and our proposed ILRED algorithms is the error correction branch, this performance improvement demonstrates the effect of the error correction branch in our proposed algorithm on dynamic point clouds. One example of the spectral result using ILRED algorithms is shown in Fig. 8, where the cluster error points are labeled in pink. As shown in Fig. 8(c) and (d), our proposed ILRED algorithms can keep the majority of the clustering results to be the same as the ground truth.

2) Large-scale Point Cloud: In real-world applications, point clouds may contain millions of points [30]. Traditional ED algorithms need an enormous amount of memory to process such a large-scale matrix directly, which makes the

Fig. 9. Boxer point cloud in 8i Voxelized Surface Light Field (8iVSLF) Dataset.

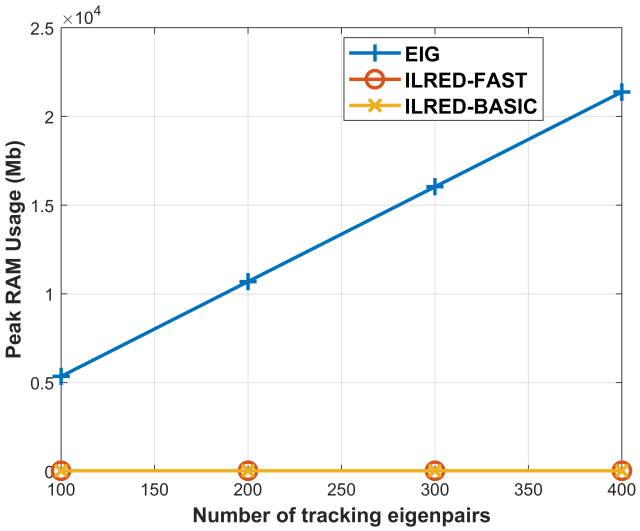

Fig. 10. Peak memory usage of the ED algorithms on Boxer point cloud.

process of estimating its eigen-pairs very difficult. However, our proposed ILRED algorithms only require a small amount of memory, which provides a solution for these problems. In this experiment, we test a simple algorithm on estimating the top K eigen-pairs of the adjacency matrix of a large-scale point cloud. We first select a part of the large-scale point cloud containing N_0 points. Subsequently, we build the adjacency matrix on this part based on its intrinsic distance with the Gaussian kernel. Then we calculate the top K eigenpairs of this adjacency matrix as the input of the ILRED-FAST algorithm. Next, we add one point in the original point cloud each time to the existing part, while using the same intrinsic distance value to calculate the corresponding additional column vector $\boldsymbol{\alpha}$ and d in the updated adjacency matrix. $\boldsymbol{\alpha}$ and d are used in the ILRED-FAST algorithm to update the approximation of the top K eigenpairs. To demonstrate the memory efficiency of our proposed algorithm, we test it on the Boxer point cloud in 8i Voxelized Surface Light Field (8iVSLF) Dataset [29], which contains about 3.49 million nodes, as shown in Fig. 9. We measure the peak memory usage by using the built-in profiler in Matlab. The peak memory usage result is shown in Fig. 10. Compared with the conventional EIG algorithm, our proposed ILRED-FAST has much lower peak memory usage, since we can free the memory of the adjacency matrix once initial eigenpairs are calculated.

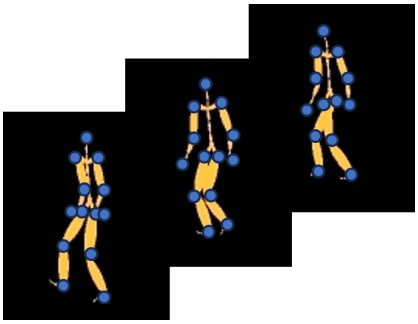

Fig. 11. Example of motion capture data.

3) Motion Capture Data: Human motion analysis has recently emerged as an active research field, stemming from its broad applications in many areas, ranging from human-robot interaction to autonomous driving [31]. Motion capture data contains a sequence of data frames captured by sensors mounted on the human body, as shown in Fig. 11. If we consider the attributes of all nodes in one frame as the signal on one vertex, we can build a temporal correlation matrix on motion capture data. With the number of data frames increasing throughout time, the size of the temporal correlation matrix increases as well, which provides a background for ED algorithms. One attractive problem on motion capture data is motion segmentation, which aims to separate the motion capture sequence into segments, each corresponding to one kind of motion. Of all existing works on motion segmentation, spectral clustering is one of the basic methods. The authors in [38] used spectral clustering as the basic comparison in their experiments. On the other hand, many works develop new clustering methods based on spectral clustering. The authors in [39] employed two methods of spectral clustering, t-nearest neighbors and the Nystrom method, to cluster motion capture data for getting behavioral segmentation. Spectral clustering was also used in [40] as a part of the proposed framework of sparse subspace clustering based on Riemannian manifold structure. Graph filters are also used in motion data analysis. Irregular-aware graph filters and graph Fourier transform were proposed in [41] by considering the irregular relationships between the data points on applications like motion capture data.

In this part, we test the ED algorithms on trails 01 to 14 of subject 86 in CMU graphics lab motion capture database[1]. Similar to Section V-B1, we conduct 10 Monte Carlo runs on each trail. In each run, the temporal correlation matrix is based by the L_2 distance between all data values of two frames, and uses a Gaussian kernel to calculate the correlation value. This correlation matrix is the final matrix of all ED algorithms. Next, we randomly drop 1% of the frames in the wireframe sequence and remove its corresponding rows and columns in the correlation matrix. We then calculate the top K eigenpairs of this truncated correlation matrix to generate the input decomposition matrix $\mathbf{Q}_{\text{init}}$ and $\boldsymbol{\Sigma}_{\text{init}}$. In our test, K is still set to 100. After getting the approximated ED results $\mathbf{Q}$ and $\boldsymbol{\Sigma}$ of the final matrix, we use them to do spectral clustering, Haar-

[1]http://mocap.cs.cmu.edu/

Algorithm 4 Fast Incremental Eigen-Decomposition Updating Algorithm for Motion Capture Data

Input: Attributes of the updated frames $\mathcal{A}_u$, adjacency matrix of the point cloud at the current frame $\mathbf{A}_{\text{init}}$, number of tracking eigen-pairs K, intrinsic resolution d_r;
Output: Decomposition matrix $\mathbf{Q}$ and diagonal eigenvalue matrix $\mathbf{\Sigma}$ as the approximated ED of the adjacency matrix of the updated matrix $\mathbf{A}_{\text{final}}$;
1. For each updated frame whose attributes are in $\mathcal{A}_u$, calculate the additional vector $\boldsymbol{\alpha}_i$ as Eq. (II.1), set all singular values $\{d_i\}$ as 1;
2. Calculate $\mathbf{Q}_{\text{init}}$ and $\mathbf{\Sigma}_{\text{init}}$ containing the top K eigen-pairs of $\mathbf{A}_{\text{init}}$ using conventional EIG algorithm;
3. Use Alg. 1 or Alg. 2 to calculate the approximated ED results $\mathbf{Q}$ and $\mathbf{\Sigma}$ for the updated frame;
4. Trim $\mathbf{Q}$ and $\mathbf{\Sigma}$ by keeping the top K eigen-pairs.

Algorithm 5 Fast Incremental Eigen-Decomposition Updating Algorithm for Hyperspectral Image

Input: Attributes of the updated columns $\mathcal{C}_u$, adjacency matrix of the point cloud at the current frame $\mathbf{A}_{\text{init}}$, number of tracking eigen-pairs K, intrinsic resolution d_r;
Output: Decomposition matrix $\mathbf{Q}$ and diagonal eigenvalue matrix $\mathbf{\Sigma}$ as the approximated ED of the adjacency matrix of the updated matrix $\mathbf{A}_{\text{final}}$;
1. For each updated point whose attributes are in $\mathcal{C}_u$, calculate the additional vector $\boldsymbol{\alpha}_i$ as Eq. (II.1), set all singular values $\{d_i\}$ as 1;
2. Calculate $\mathbf{Q}_{\text{init}}$ and $\mathbf{\Sigma}_{\text{init}}$ containing the top K eigen-pairs of $\mathbf{A}_{\text{init}}$ using conventional EIG algorithm;
3. Use Alg. 1 or Alg. 2 to calculate the approximated ED results $\mathbf{Q}$ and $\mathbf{\Sigma}$ for the updated point;
4. Trim $\mathbf{Q}$ and $\mathbf{\Sigma}$ by keeping the top K eigen-pairs.

TABLE III
AVERAGE CLUSTERING ERRORS AND LOW-PASS FILTER ERRORS ON MOTION CAPTURE DATA

	ILRED-FAST	ILRED-BASIC	EEU	TRIP-BASIC	TRIP
Number of Clustering Errors	**8.2143**	8.3571	22.1429	65.1244	63.1054
L2 (Haar)	0.4568	**0.4566**	0.5810	0.8157	0.7982
L2 (Ideal)	0.4569	**0.4567**	0.5812	0.8136	0.8065

TABLE IV
AVERAGE CLUSTERING ERRORS AND LOW-PASS FILTER ERRORS ON HYPERSPECTRAL IMAGES

	ILRED-FAST	ILRED-BASIC	EEU	TRIP-BASIC	TRIP
Number of Clustering Errors	**70.5**	**70.5**	71.5	1529.5	1875.6
L2 (Haar)	13820	**13780**	13918	55670	62845
L2 (Ideal)	14756	**14670**	14872	75525	84772

like low-pass filtering and ideal low-pass filtering compare them with the ground truth results using the top K eigen-pairs of the full-sized correlation matrix. We summarize this ED updating algorithm in Alg. 4.

The average result is shown in Table. III. Our proposed ILRED algorithms have the lowest error in all three metrics, which demonstrates the accuracy of our proposed algorithm on motion capture data.

4) Hyperspectral Image: Hyperspectral image has been widely used in applications like earth observation and industrial scanning [32]. Hyperspectral image contains images of multiple spectral bands on the same object. The pixels in hyperspectral image can be viewed as the vertices of a graph, while the values of all bandwidth form attributes on the vertices. One common sensor of capturing the hyperspectral image is called pushbroom hyperspectral imager, which scans a line at each moment in time. Therefore, the size of the hyperspectral image is increasing over time, and this scenario is suitable for the ED algorithms. Spectral clustering and graph-based filters are widely used on hyperspectral images. A fast spectral clustering was proposed in [42] for unsupervised hyperspectral image classification. The authors in [43] proposed a spatial-spectral clustering with anchor graph for HSI data clustering. On the other hand, the authors in [44] designed a linear function to combine the different graph filters in their proposed framework so that the graph filter can be adaptively determined by training different weight matrices.

In this part, we test the ED algorithms on the Indian Pines data [45]. We follow a similar workflow as Section V-B1 and Section V-B3. The major difference of our experiment for hyperspectral image is that we drop the last 5% columns of the original hyperspectral image to simulate the real situation of using a pushbroom hyperspectral imager. We first build the ϵ-neighborhood graph on the hyperspectral image with the radius of neighborhood as three times of the intrinsic distance, where the intrinsic distance is defined as the mean of the distances between all pixels and its nearest neighbor. The distance between two pixels is measured by the L_2 norm of the difference on attributes of these two pixels. Then we calculate the adjacency matrix of the constructed graph using the Gaussian kernel. This adjacency matrix is the final matrix of all ED algorithms. Next, we drop the last 5% columns of the original hyperspectral image, and remove its corresponding rows and columns in the adjacency matrix. We then calculate the top K eigen-pairs of this truncated adjacency matrix to generate the input decomposition matrix $\mathbf{Q}_{\text{init}}$ and $\mathbf{\Sigma}_{\text{init}}$. In our test, K is set to 100. We summarize this ED updating algorithm in Alg. 5.

After getting the approximated ED results $\mathbf{Q}$ and $\mathbf{\Sigma}$ of the final matrix, we use them for spectral clustering, Haar-like low-pass filtering and ideal low-pass filtering compare them with the ground truth results using the top K eigen-pairs of the full-sized adjacency matrix. For spectral clustering results, the error performance is measured by the number of pixels that are incorrectly clustered. For low-pass filtering results, the error performance is measured by the L_2 norm between the filtered signals using approximated results and ground truth.

The average result is shown in Table. IV. Our proposed ILRED algorithms have the lowest error in all three metrics, which demonstrates the accuracy of our proposed algorithm

on hyperspectral images.

VI. Conclusion

This work studies the ED approximation algorithms for low-rank matrices in GSP with incrementally-updated graph structure. Specially, we develop a basic ILRED algorithm with the error correction branch to improve the estimation accuracy. We also proposed a faster ILRED-FAST by introducing a product decomposition form. Our proposed methods are easier to implement, and show robustness across both synthetic and real-world datasets. Our experimental results demonstrated the efficacy on runtime and memory, as well as the accuracy of eigenvalue calculation. This work establishes ILRED as an efficient tool to approximate low-rank ED on dynamic point clouds, motion capture data and hyperspectral images. In future works, we plan to extend this algorithm to process shrinking (decremental) graphs and high-order tensors in high-dimensional graph signal processing [46], [47]. We also view the fast ED algorithm on sparse symmetric matrices with higher rank as another promising direction for exploration.

References

[1] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura and P. Vandergheynst, "Graph Signal Processing: Overview, Challenges, and Applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808-828, May 2018.

[2] W. Hu, J. Pang, X. Liu, D. Tian, C. -W. Lin and A. Vetro, "Graph Signal Processing for Geometric Data and Beyond: Theory and Applications," *IEEE Transactions on Multimedia*, vol. 24, pp. 3961-3977, 2022.

[3] R. Li et al., "Graph Signal Processing, Graph Neural Network and Graph Learning on Biological Data: A Systematic Review," *IEEE Reviews in Biomedical Engineering*, vol. 16, pp. 109-135, 2023.

[4] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: frequency analysis," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3042-3054, Jun., 2014.

[5] A. Sandryhaila, and J. M. F. Moura, "Discrete signal processing on graphs: graph filters," in *Proceedings of 2013 IEEE ICASSP*, Vancouver, Canada, May 2013, pp. 6163-6166.

[6] Q. Zhang, Y. Tian, T. Wang, F. Yuan and Q. Xu, "ApproxEigen: An approximate computing technique for large-scale eigen-decomposition," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Austin, TX, USA, 2015, pp. 824-830.

[7] U. Von Luxburg, "A tutorial on spectral clustering", Stat. Comput., vol. 17, no. 4, pp. 395-416, 2007.

[8] J. Liu, C. Wang, M. Danilevsky and J. Han, "Large-scale spectral clustering on graphs", Proc. Int. Joint Conf. Artif. Intell., pp. 1486-1492, 2013.

[9] H. Yin, W. Hu, Z. Zhang, J. Lou and M. Miao, "Incremental multi-view spectral clustering with sparse and connected graph learning", Neural Netw., vol. 144, pp. 260-270, Dec. 2021.

[10] K. . -B. Yu, "Recursive updating the eigenvalue decomposition of a covariance matrix," *IEEE Transactions on Signal Processing*, vol. 39, no. 5, pp. 1136-1145, May 1991.

[11] R. D. DeGroat and R. A. Roberts, "Efficient, numerically stabilized rank-one eigenstructure updating (signal processing)," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 2, pp. 301-316, Feb. 1990.

[12] R. Bru, R. Canto, and A. M. Urbano, "Eigenstructure of rank one updated matrices," *Linear Algebra and its Applications*, vol. 485, pp. 372-391, 2015.

[13] H. Ning, W. Xu, Y. Chi, Y. Gong, and T. Huang, "Incremental spectral clustering with application to monitoring evolving blog communities," in *SIAM International Conference on Data Mining*, Citeseer, 2007.

[14] H. Ning, W. Xu, Y. Chi, Y. Gong, and T. S. Huang, "Incremental spectral clustering by efficiently updating the eigen-system," *Pattern Recognit.*, vol. 43, no. 1, pp. 113-127, 2010.

[15] C. Dhanjal, R. Gaudel, and S. Clémençon, "Efficient eigen-updating for spectral graph clustering," *Neurocomputing*, vol. 131, pp. 440-452, May 2014.

[16] J. T. Kwok and H. Zhao, "Incremental eigen decomposition," in *Proc. ICANN*, Istanbul, Turkey, Jun. 2003, pp. 270-273.

[17] C. Chen and H. Tong, "Fast eigen-functions tracking on dynamic graphs," in *Proc. SIAM Int. Conf. Data Mining*, 2015, pp. 559-567.

[18] G. W. Stewart and J.-G. Sun, *Matrix Perturbation Theory*. Boston, MA, USA: Academic, 1990.

[19] H. E. Egilmez and A. Ortega, "Spectral anomaly detection using graph-based filtering for wireless sensor networks," in *Proceedings of 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, 2014, pp. 1085-1089.

[20] J.D. Medaglia, W. Huang, E.A. Karuza, et al., "Functional alignment with anatomical networks is associated with cognitive flexibility," *Nature Human Behaviour*, vol. 2, pp. 156-164, 2018.

[21] W. Huang, T. A. W. Bolton, J. D. Medaglia, D. S. Bassett, A. Ribeiro and D. Van De Ville, "A Graph Signal Processing Perspective on Functional Brain Imaging," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 868-885, May 2018.

[22] U. von Luxburg, "A tutorial on spectral clustering," *Stat. Comput.*, vol. 17, no. 4, pp. 395-416, 2007.

[23] T. Kong, Y. Tian and H. Shen, "A Fast Incremental Spectral Clustering for Large Data Sets," in *2011 12th International Conference on Parallel and Distributed Computing, Applications and Technologies*, Gwangju, Korea (South), 2011, pp. 1-5.

[24] S. Chen, B. Liu, C. Feng, C. Vallespi-Gonzalez and C. Wellington, "3D Point Cloud Processing and Learning for Autonomous Driving: Impacting Map Creation, Localization, and Perception," *IEEE Signal Processing Magazine*, vol. 38, no. 1, pp. 68-86, Jan. 2021.

[25] W. Zhu, Z. Ma, Y. Xu, L. Li and Z. Li, "View-Dependent Dynamic Point Cloud Compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 2, pp. 765-781, Feb. 2021.

[26] D. Rempe, T. Birdal, Y. Zhao, Z. Gojcic, S. Sridhar, and L. J. Guibas, "CaSPR: Learning canonical spatiotemporal point cloud representations," in *Proc. Conf. Neural Inf. Process. Syst.*, 2020, pp. 13688-13701.

[27] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, "Numerical geometry of non-rigid shapes," Springer, 2008.

[28] A. M. Bronstein, M. M. Bronstein, A. M. Bruckstein, and R. Kimmel, "Analysis of two-dimensional non-rigid shapes," *Intl. J. Computer Vision (IJCV)*, vol. 78, no. 1, pp. 67-88, June 2008.

[29] M. Krivokuća, P. A. Chou, and P. Savill, "8i Voxelized Surface Light Field (8iVSLF) Dataset," ISO/IEC JTC1/SC29 WG11 (MPEG) input document m42914, Ljubljana, July 2018.

[30] Q. Deng, S. Zhang and Z. Ding, "An Efficient Hypergraph Approach to Robust Point Cloud Resampling," in IEEE Transactions on Image Processing, vol. 31, pp. 1924-1937, 2022.

[31] Y. Desmarais, D. Mottet, P. Slangen and P. Montesinos, "A review of 3d human pose estimation algorithms for markerless motion capture," *Computer Vision and Image Understanding*, vol. 212, Nov. 2021.

[32] V. Lodhi, D. Chakravarty, and P. Mitra, "Hyperspectral imaging system: Development aspects and recent trends," *Sensing and Imaging*, vol. 20, pp. 1-24, 2019.

[33] G. Ranjan, Z. L. Zhang, and D. Boley, "Incremental computation of pseudo-inverse of laplacian," in *International Conference on Combinatorial Optimization and Applications*, Cham: Springer International Publishing, Nov. 2014, pp. 729-749.

[34] G. Bravo-Hermsdorff and L. M. Gunderson. "A unifying framework for spectrum-preserving graph sparsification and coarsening," *Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[35] C. Jiang, D. P. Paudel, D. Fofi, Y. Fougerolle and C. Demonceaux, "Moving Object Detection by 3D Flow Field Analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 1950-1963, April 2021.

[36] D. Wang et al., "Separating tree photosynthetic and non-photosynthetic components from point cloud data using dynamic segment merging," *Forests*, vol. 9, no. 5, p. 252, 2018.

[37] R. Watanabe, K. Nonaka, E. Pavez, T. Kobayashi and A. Ortega, "Graph-Based Point Cloud Color Denoising with 3-Dimensional Patch-Based Similarity," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Rhodes Island, Greece, 2023, pp. 1-5.

[38] F. Zhou, F. De la Torre and J. K. Hodgins, "Hierarchical Aligned Cluster Analysis for Temporal Clustering of Human Motion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 3, pp. 582-596, Mar. 2013.

[39] X. Yu, W. Liu, and W. Xing, "Behavioral segmentation for human motion capture data based on graph cut method," *J. Vis. Lang. Comput.*, vol. 43, pp. 50-59, Dec. 2017.

[40] G. Xia, H. Sun, L. Feng, G. Zhang and Y. Liu, "Human Motion Segmentation via Robust Kernel Sparse Subspace Clustering," *IEEE Transactions on Image Processing*, vol. 27, no. 1, pp. 135-150, Jan. 2018.
[41] B. Girault, A. Ortega and S. S. Narayanan, "Irregularity-Aware Graph Fourier Transforms," *IEEE Transactions on Signal Processing*, vol. 66, no. 21, pp. 5746-5761, 1 Nov.1, 2018.
[42] Y. Zhao, Y. Yuan, and Q. Wang, "Fast spectral clustering for unsupervised hyperspectral image classification," *Remote Sens.*, vol. 11, no. 4, pp. 399, Feb. 2019.
[43] Q. Wang, Y. Miao, M. Chen and Y. Yuan, "Spatial-Spectral Clustering With Anchor Graph for Hyperspectral Image," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1-13, 2022.
[44] Y. Ding et al., "AF2GNN: Graph convolution with adaptive filters and aggregator fusion for hyperspectral image classification," *Inf. Sci.*, vol. 602, pp. 201-219, Jul. 2022.
[45] M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe, "220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3," *Purdue University Research Repository*, 2015.
[46] S. Zhang, Z. Ding and S. Cui, "Introducing Hypergraph Signal Processing: Theoretical Foundation and Practical Applications," in IEEE Internet of Things Journal, vol. 7, no. 1, pp. 639-660, Jan. 2020.
[47] S. Zhang, Q. Deng and Z. Ding, "Signal Processing Over Multilayer Graphs: Theoretical Foundations and Practical Applications," in IEEE Internet of Things Journal, vol. 11, no. 2, pp. 2453-2471, 15 Jan.15, 2024.

APPENDIX

A. Proof of Theorem 1

The right-hand side of Eq. (III.8) can be written as

$$\begin{aligned}
&\begin{bmatrix} \mathbf{Q} & \widetilde{\mathbf{e}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{\Sigma} & 0 & \mathbf{Q}^\top \boldsymbol{\alpha} \\ 0 & 0 & p \\ \boldsymbol{\alpha}^\top \mathbf{Q} & p & d \end{bmatrix} \begin{bmatrix} \mathbf{Q} & \widetilde{\mathbf{e}} & 0 \\ 0 & 0 & 1 \end{bmatrix}^\top \\
&= \begin{bmatrix} \mathbf{Q}\mathbf{\Sigma} & 0 & \mathbf{Q}\mathbf{Q}^\top \boldsymbol{\alpha} + \widetilde{\mathbf{e}} p \\ \boldsymbol{\alpha}^\top \mathbf{Q} & p & d \end{bmatrix} \begin{bmatrix} \mathbf{Q}^\top & 0 \\ \widetilde{\mathbf{e}}^\top & 0 \\ 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top & \mathbf{Q}\mathbf{Q}^\top \boldsymbol{\alpha} + \mathbf{e} \\ \boldsymbol{\alpha}^\top \mathbf{Q}\mathbf{Q}^\top + \mathbf{e}^\top & d \end{bmatrix}
\end{aligned} \tag{A.1}$$

Substitute Eq. (III.3) into Eq. (A.1), we have

$$\begin{aligned}
&\begin{bmatrix} \mathbf{Q} & \widetilde{\mathbf{e}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{\Sigma} & 0 & \mathbf{Q}^\top \boldsymbol{\alpha} \\ 0 & 0 & p \\ \boldsymbol{\alpha}^\top \mathbf{Q} & p & d \end{bmatrix} \begin{bmatrix} \mathbf{Q} & \widetilde{\mathbf{e}} & 0 \\ 0 & 0 & 1 \end{bmatrix}^\top \\
&= \begin{bmatrix} \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top & \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^\top & d \end{bmatrix} \\
&= \mathbf{M}_{\text{new}},
\end{aligned} \tag{A.2}$$

which completes the proof.

B. Proof of Theorem 2

Let $\widetilde{\mathbf{M}_{\text{new}}} = \begin{bmatrix} \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top & \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^\top & d \end{bmatrix}$ and we split the proof into the following two cases:

Case 1: $p \le \varepsilon$. Recall that $\mathbf{Y}_1 = \begin{bmatrix} \mathbf{\Sigma} & \mathbf{Q}^\top \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^\top \mathbf{Q} & d \end{bmatrix}$ and let $\mathbf{Q}_1 \mathbf{\Sigma_1} \mathbf{Q}_1^\top$ be the eigen-decomposition of $\mathbf{Y}_1$, let λ_1 be the smallest eigenvalue of $\mathbf{Y}_1$.

(1) If $\lambda_1 > \varepsilon_\lambda$, then from the Alg. 1 we will not apply eigenvalue truncation, i.e,

$$\mathbf{Q}_{\text{new}} = \begin{bmatrix} \mathbf{Q} & 0 \\ 0 & 1 \end{bmatrix} \mathbf{Q}_1, \qquad \mathbf{\Sigma}_{\text{new}} = \mathbf{\Sigma_1}.$$

Direct computation gives

$$\mathbf{Q}_{\text{new}} \mathbf{\Sigma}_{\text{new}} \mathbf{Q}_{\text{new}}^\top = \begin{bmatrix} \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top & \mathbf{Q}\mathbf{Q}^\top \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^\top \mathbf{Q}\mathbf{Q}^\top & d \end{bmatrix}.$$

Therefore,

$$\begin{aligned}
&\mathbf{M}_{\text{new}} - \mathbf{Q}_{\text{new}} \mathbf{\Sigma}_{\text{new}} \mathbf{Q}_{\text{new}}^\top \\
&= \begin{bmatrix} \mathbf{M}_{\text{old}} & \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^\top & d \end{bmatrix} - \begin{bmatrix} \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top & \mathbf{Q}\mathbf{Q}^\top \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^\top \mathbf{Q}\mathbf{Q}^\top & d \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{M}_{\text{old}} - \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top & \boldsymbol{\alpha} - \mathbf{Q}\mathbf{Q}^\top \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^\top - \boldsymbol{\alpha}^\top \mathbf{Q}\mathbf{Q}^\top & 0 \end{bmatrix}.
\end{aligned}$$

This implies

$$\begin{aligned}
&\left\| \mathbf{M}_{\text{new}} - \mathbf{Q}_{\text{new}} \mathbf{\Sigma}_{\text{new}} \mathbf{Q}_{\text{new}}^\top \right\|_F^2 \\
&= \left\| \mathbf{M}_{\text{old}} - \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top \right\|_F^2 + 2\|\boldsymbol{\alpha} - \mathbf{Q}\mathbf{Q}^\top \boldsymbol{\alpha}\|^2 \\
&= \left\| \mathbf{M}_{\text{old}} - \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top \right\|_F^2 + 2p^2.
\end{aligned}$$

Hence,

$$\left\| \mathbf{M}_{\text{new}} - \mathbf{Q}_{\text{new}} \mathbf{\Sigma}_{\text{new}} \mathbf{Q}_{\text{new}}^\top \right\|_F \le \left\| \mathbf{M}_{\text{old}} - \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top \right\|_F + \sqrt{2}p.$$

(2) If $\lambda_1 \le \varepsilon_\lambda$, then from Alg. 1, we have the eigenvalue truncation as

$$\mathbf{Q}_{\text{new}} = \begin{bmatrix} \mathbf{Q} & 0 \\ 0 & 1 \end{bmatrix} \mathbf{Q}_1(:, 1:k), \qquad \mathbf{\Sigma}_{\text{new}} = \mathbf{\Sigma}_1(1:k, 1:k).$$

Assuming that $\mathbf{q}$ is the eigenvector of $\widetilde{\mathbf{M}_{\text{new}}}$, which corresponds to the eigenvalue λ_1. Then we have

$$\mathbf{Q}_{\text{new}} \mathbf{\Sigma}_{\text{new}} \mathbf{Q}_{\text{new}}^\top = \begin{bmatrix} \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top & \mathbf{Q}\mathbf{Q}^\top \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^\top \mathbf{Q}\mathbf{Q}^\top & d \end{bmatrix} - \lambda_1 \mathbf{q}\mathbf{q}^\top.$$

Therefore,

$$\begin{aligned}
&\mathbf{M}_{\text{new}} - \mathbf{Q}_{\text{new}} \mathbf{\Sigma}_{\text{new}} \mathbf{Q}_{\text{new}}^\top \\
&= \begin{bmatrix} \mathbf{M}_{\text{old}} & \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^\top & d \end{bmatrix} - \begin{bmatrix} \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top & \mathbf{Q}\mathbf{Q}^\top \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^\top \mathbf{Q}\mathbf{Q}^\top & d \end{bmatrix} + \lambda_1 \mathbf{q}\mathbf{q}^\top \\
&= \begin{bmatrix} \mathbf{M}_{\text{old}} - \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top & \boldsymbol{\alpha} - \mathbf{Q}\mathbf{Q}^\top \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^\top - \boldsymbol{\alpha}^\top \mathbf{Q}\mathbf{Q}^\top & 0 \end{bmatrix} + \lambda_1 \mathbf{q}\mathbf{q}^\top.
\end{aligned}$$

This implies

$$\begin{aligned}
&\left\| \mathbf{M}_{\text{new}} - \mathbf{Q}_{\text{new}} \mathbf{\Sigma}_{\text{new}} \mathbf{Q}_{\text{new}}^\top \right\|_F \\
&\le \left\| \mathbf{M}_{\text{old}} - \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top \right\|_F + \sqrt{2}\|\boldsymbol{\alpha} - \mathbf{Q}\mathbf{Q}^\top \boldsymbol{\alpha}\| + \lambda_{\min} \\
&= \left\| \mathbf{M}_{\text{old}} - \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top \right\|_F + \sqrt{2}p + \lambda_1.
\end{aligned}$$

Case 2: $p > \varepsilon$. Recall that

$$\mathbf{Y}_2 = \begin{bmatrix} \mathbf{\Sigma} & 0 & \mathbf{Q}^\top \boldsymbol{\alpha} \\ 0 & 0 & p \\ \boldsymbol{\alpha}^\top \mathbf{Q} & p & d \end{bmatrix},$$

where $\widetilde{\mathbf{e}} = \mathbf{e}/p$ and $\mathbf{Q}_2 \mathbf{\Sigma_2} \mathbf{Q}_2^\top$ is the eigen-decomposition of $\mathbf{Y}_2$, let λ_2 be the smallest eigenvalue of $\mathbf{Y}_2$.

(1) If $\lambda > \varepsilon_\lambda$, then from Alg. 1 we have

$$\mathbf{Q}_{\text{new}} = \begin{bmatrix} \mathbf{Q} & \widetilde{\mathbf{e}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{Q}_2, \qquad \mathbf{\Sigma}_{\text{new}} = \mathbf{\Sigma}_2,$$

Direct computation gives

$$\mathbf{Q}_{\text{new}}\mathbf{\Sigma}_{\text{new}}\mathbf{Q}_{\text{new}}^{\top} = \begin{bmatrix} \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^{\top} & \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^{\top} & d \end{bmatrix}.$$

Therefore,

$$\begin{aligned}
&\mathbf{M}_{\text{new}} - \mathbf{Q}_{\text{new}}\mathbf{\Sigma}_{\text{new}}\mathbf{Q}_{\text{new}}^{\top} \\
&= \begin{bmatrix} \mathbf{M}_{\text{old}} & \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^{\top} & d \end{bmatrix} - \begin{bmatrix} \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^{\top} & \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^{\top} & d \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{M}_{\text{old}} - \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^{\top} & 0 \\ 0 & 0 \end{bmatrix}.
\end{aligned}$$

This implies

$$\left\|\mathbf{M}_{\text{new}} - \mathbf{Q}_{\text{new}}\mathbf{\Sigma}_{\text{new}}\mathbf{Q}_{\text{new}}^{\top}\right\|_F = \left\|\mathbf{M}_{\text{old}} - \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^{\top}\right\|_F.$$

(2) If $\lambda_2 \le \varepsilon_\lambda$, then from Alg. 1 we have

$$\mathbf{Q}_{\text{new}} = \begin{bmatrix} \mathbf{Q} & \widetilde{e} & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{Q}_2(:, 1:k), \qquad \mathbf{\Sigma}_{\text{new}} = \mathbf{\Sigma}_2(1:k, 1:k).$$

Direct computation gives

$$\mathbf{Q}_{\text{new}}\mathbf{\Sigma}_{\text{new}}\mathbf{Q}_{\text{new}}^{\top} = \begin{bmatrix} \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^{\top} & \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^{\top} & d \end{bmatrix} - \lambda_2 \mathbf{q}\mathbf{q}^{\top},$$

$\mathbf{q}$ is the eigenvector of $\widetilde{\mathbf{M}_{\text{new}}}$ which corresponding to the eigenvalue λ_2. Therefore,

$$\begin{aligned}
&\mathbf{M}_{\text{new}} - \mathbf{Q}_{\text{new}}\mathbf{\Sigma}_{\text{new}}\mathbf{Q}_{\text{new}}^{\top} \\
&= \begin{bmatrix} \mathbf{M}_{\text{old}} & \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^{\top} & d \end{bmatrix} - \begin{bmatrix} \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^{\top} & \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^{\top} & d \end{bmatrix} + \lambda_2 \mathbf{q}\mathbf{q}^{\top} \\
&= \begin{bmatrix} \mathbf{M}_{\text{old}} - \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^{\top} & 0 \\ 0 & 0 \end{bmatrix} + \lambda_2 \mathbf{q}\mathbf{q}^{\top}.
\end{aligned}$$

This implies

$$\begin{aligned}
&\left\|\mathbf{M}_{\text{new}} - \mathbf{Q}_{\text{new}}\mathbf{\Sigma}_{\text{new}}\mathbf{Q}_{\text{new}}^{\top}\right\|_F \\
&\le \left\|\mathbf{M}_{\text{old}} - \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^{\top}\right\|_F + \|\lambda_2 \mathbf{q}\mathbf{q}^{\top}\|_F \\
&\le \left\|\mathbf{M}_{\text{old}} - \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^{\top}\right\|_F + \lambda_2.
\end{aligned}$$

C. Proof of Theorem 3

From Theorem 2, we know that when $p \le \varepsilon$, the p-truncation is applied and the eigen approximation error is increased by no larger than $\sqrt{2}p$. Since $p \le \varepsilon$, such eigen approximation error increase is also no larger than $\sqrt{2}\varepsilon$. On the other hand, when the eigenvalue truncation is applied, the eigen approximation error is increased by no larger than λ_1 or λ_2. Given that both λ_1 and λ_2 are no larger than ε_λ in this case, eigen approximation error increase is no larger than ε_λ when the eigenvalue truncation is applied. Suppose T_p represents the total number of times p-truncation is applied and T_λ represents the total number of times the eigenvalue truncation is applied, then the total accumulated eigen approximation error

$$\left\|\mathbf{M}_{\text{old}} - \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^{\top}\right\|_F \le \sqrt{2}T_p\varepsilon + T_\lambda\varepsilon_\lambda.$$

D. Proof of Theorem 4

From the analysis in Section IV-B1, the complexity order of the iteration in updating the initial matrix $\mathbf{M}_{\text{old}} \in \mathbb{R}^{n\times n}$ with rank k to $\mathbf{M}_{\text{new}} \in \mathbb{R}^{(n+1)\times(n+1)}$ is $O(nk^2 + k^3)$. In the worst scenario, all ℓ iterations of updating will always increase the rank of the updated matrix. Therefore, the overall complexity of ILRED-BASIC algorithm is bounded by the order of

$$\begin{aligned}
&\sum_{i=0}^{\ell-1}(n+i)(k+i)^2 + (k+i)^3 \\
&= \frac{5}{6}k\ell + \frac{1}{6}n\ell - \frac{5}{2}k\ell^2 - 2k^2\ell + \frac{5}{3}k\ell^3 + k^3\ell - \frac{1}{2}n\ell^2 \\
&+ \frac{1}{3}n\ell^3 + \frac{1}{2}\ell^2 - \ell^3 + \frac{1}{2}\ell^4 + 2k^2\ell^2 + kn\ell^2 + k^2n\ell - kn\ell
\end{aligned} \tag{A.3}$$

By ignoring the constant factors and lower order terms, the overall complexity of ILRED-BASIC algorithm is $O(k\ell^3 + k^2\ell^2 + k^3\ell + kn\ell^2 + k^2n\ell + n\ell^3 + \ell^4)$.

E. Proof of Theorem 5

From the analysis in Section IV-B2, the complexity order of the iteration in updating the initial matrix $\mathbf{M}_{\text{old}} \in \mathbb{R}^{n\times n}$ with rank k to $\mathbf{M}_{\text{new}} \in \mathbb{R}^{(n+1)\times(n+1)}$ is $O(nk + k^3)$. In the worst scenario, all ℓ iterations of updating will always increase the rank of the updated matrix. Therefore, the overall complexity of ILRED-FAST algorithm is bounded by the order of

$$\begin{aligned}
&\sum_{i=0}^{\ell-1}(n+i)(k+i) + (k+i)^3 \\
&= \frac{1}{6}\ell - \frac{1}{2}n\ell - k\ell^2 - \frac{3}{2}k^2\ell + k\ell^3 + k^3\ell + \frac{1}{2}n\ell^2 - \frac{1}{4}\ell^2 \\
&- \frac{1}{6}\ell^3 + \frac{1}{4}\ell^4 + \frac{3}{2}k^2\ell^2 + kn\ell
\end{aligned} \tag{A.4}$$

By ignoring the constant factors and lower order terms, the overall complexity of ILRED-BASIC algorithm is $O(k\ell^3 + k^2\ell^2 + k^3\ell + \ell^4 + n\ell^2 + kn\ell)$.