An Investigation into Spectral Sequencing using Graph Distance

Rong Liu, Hao Zhang, and Oliver van Kaick

GrUVi Lab, School of Computing Sciences,
Simon Fraser University, Burnaby, British Columbia, Canada
{lrong, haoz, ovankaick}@cs.sfu.ca

Technical Report TR 2006-08
School of Computing Science
Simon Fraser University

Abstract. The construction of linear mesh layouts has found various applications, such as implicit mesh filtering and mesh streaming, where a variety of layout quality criteria, e.g., width and span, can be considered. Similar linear sequencing problems have also been studied in the context of sparse matrix reordering and graph labeling, where width and span correspond to vertex separation and bandwidth, respectively. One of the best-known heuristics for generating width-minimizing orderings is spectral sequencing, which is derived from the Fiedler vector. In terms of span however, other heuristics, such as the Cuthill-Mckee (CM) scheme, generally outperform spectral sequencing. In this paper, we study the general linear sequence generation as a problem of preserving graph distances and propose to use for sequencing the subdominant eigenvector of a kernel (affinity) matrix, defined by graph distances and appropriately chosen transfer functions. The use of Laplacians can then be seen as a special case, where a step transfer function of unit width is applied. Despite the non-sparsity of the kernel matrix we use, the sequences can be computed efficiently for problems of large size through subsampling and eigenvector extrapolation. When applied to mesh layouts generation, we show experimentally that the sequences obtained using our algorithm outperform those derived from the Fiedler vector, in terms of spans, and those obtained from CM, in terms of widths and other important quality criteria. Therefore, in applications where several such quality criteria can influence algorithm performance simultaneously, e.g., mesh streaming and implicit mesh filtering, the new mesh layouts could potentially provide a better trade-off.

1 Introduction

The computation of linear mesh layouts is an instance of the general graph layout problem [1], where an optimal labeling or ordering of the vertices of a given graph is sought. Many optimization problems, from diverse fields, can be

formulated as graph layout problems; these include sparse matrix reordering in numerical analysis [2–5], circuit layout in VLSI design [6], DNA sequencing in computational biology [7], archaeological dating, as well as ranking problems in geography [8] and information retrieval [9]. A variety of layout costs have been considered and most of them lead to NP-hard optimization problems [1].

The mesh layout problem is of interest since the application-based optimal sequencing of the mesh faces, vertices, or higher-order entities, e.g., nodes in a multiresolution or bounding volume hierarchy [10], can lead to superior performance for rendering [11, 10] and geometry processing [10, 12–14], typically without modifying the run-time algorithm. Such reordering can also make large mesh data more streamable [15] and facilitate stream processing, e.g., in mesh simplification [16]. Layout costs relevant to these applications include *width* and *span* [15] for mesh streaming, matrix *profile*, *workbound*, and *bandwidth* [2, 4, 3] for solution of sparse linear systems related to mesh processing, e.g., in implicit mesh fairing [13, 14], as well as *average cache miss ratio* (ACMR) for rendering [11]. A variety of heuristics, including spectral sequencing [2, 4], breadth-first search and its variants [3, 10, 16], space-filling curves [11], and multi-level schemes via optimized local permutations [10, 17], have been proposed.

We notice that most of the abovementioned layout costs are related to how well the adjacency information in the mesh or model hierarchy is reflected in the linear layout, to ensure global coherence and locality preservation. Therefore, we propose a new spectral sequencing algorithm based on the graph distances to preserve locality information, as a solution to minimize the various layout costs, including width and span, without being limited to any specific criterion. It is worth pointing out that since the ACMR is defined on a different type of criterion and its goal is very different, it is not a concern of and addressed by our algorithm. The basic idea of our approach is to compute, from within a high dimensional feature space which provides a spatial embedding of the mesh graph vertices, a distance-preserving 1-D projection and use this projection to derive the ordering. Our algorithm is analyzed in the context of kernel principal component analysis (Kernel PCA), which is, to the best of our knowledge, a new way of studying the sequencing problem. Our analysis allows us to motivate the use of an uncentered kernel matrix (for efficiency) and its subdominant eigenvector to compute mesh layouts. The general framework we develop also sheds light on possible further improvements in spectral sequencing.

From a practical point of view, the sequences obtained with our algorithm outperform those derived from the Fiedler vector, in terms of spans, and those obtained from CM, in terms of widths, profiles, workbounds, and a few other important quality criteria. Thus in applications where several such criteria can influence algorithm performance simultaneously, e.g., mesh streaming and implicit mesh filtering, the new mesh layouts would potentially provide a better trade-off. In terms of computational cost, although the kernel matrix we arrive at is in general non-sparse, approximate eigenvector computation via subsampling and extrapolation using the Nyström method [18] allows us to compute layouts

for large meshes efficiently. Compared to algebraic multi-grid methods, such as ACE [19], Nyström approximation is much simpler and just as fast.

The rest of the paper is organized as follows. In Section 2, we define the graph layout problem and several layout costs and explain their relevance to some geometry processing problems. A few well-known heuristics are briefly described as well. To make the paper self-contained, we review Kernel PCA in Section 3. We then describe, in Section 4, our spectral sequencing algorithm. Various practical issues, such as subsampling for efficient layout computation, are addressed in Section 5. Experimental results are given in Section 6. Finally, we conclude and suggest possible future work.

2 The graph layout problem and heuristic algorithms

Consider a weighted graph $G = (V, E, w)$ with $V = \{v_1, \ldots, v_n\}$ the set of vertices, E the set of edges, and $w : E \to \mathbb{R}$ the edge weights. A (linear) *layout* of G is a labeling π of its vertices, $\pi : V \to \{1, 2, \ldots, n\}$. Depending on the application at hand, one seeks to find a graph layout that minimizes certain layout cost. For a real number $0 < p < \infty$, the *p-discrepancy* [8] of G with respect to a layout ϕ is defined as

$$\sigma_p(G, \pi) = \left(\sum_{uv \in E} w_{uv} |\pi(u) - \pi(v)|^p \right)^{1/p},$$

and for $p = \infty$, $\sigma_\infty(G, \pi) = \max_{uv \in E} |\pi(v) - \pi(v)|$. The ∞-discrepancy is also called the *bandwidth* of a layout. The minimum value $\sigma_p(G) = \min_\pi \sigma_p(G, \pi), 0 < p \leq \infty$, is called the *min-p-sum* of the graph G. Good bounds on $\sigma_p(G)$ are important but are nontrivial to obtain; see the survey of Mohar and Poljak [8].

Let$A$ be the graph adjacency matrix of G, then the ∞-discrepancy can be written as $\max_{1 \leq i \leq n} \max_{j<i, A_{ij} \neq 0}(i - j)$, which is seen to represent the bandwidth of the (symmetric) matrix A [3]. Two other matrix-based measures that have important implications in the solution of sparse linear systems are the *profile* or *envelope size* [2] and *workbound* [4] of a matrix, defined as follows, with $q = 1$ and $q = 2$, respectively,

$$\sum_{i=1}^{n} \max_{j<i, A_{ij} \neq 0} (i - j)^q.$$

These measures are closely related to the 1-discrepancy and the 2-discrepancy, respectively, where the graph layout is induced by the matrix ordering. The 1-discrepancy is also the cost for the well-known *minimum linear arrangement* or *MinLA* problem for graphs [17]. The MinLA problem can also be generalized to hypergraphs, which is adopted by Bogomjakov and Gotsman [11] for computing efficient rendering sequences. Another important layout cost measure is the *vertex separation* [1], defined as

$$\max_{1 \leq i \leq n} |\{\pi(u) \leq i : \exists \pi(v) > i, uv \in E\}|$$

Intuitively, it measures, at a certain point of the linear layout, the number of edges for which only one end vertex has been visited.

Several problems in mesh processing benefit from having an optimized mesh layout, where we either work with the primal graph (for vertex sequencing) or its dual graph (for face sequencing). Of particular interest in streaming meshes [15] are *span* and *width*, which correspond to bandwidth and vertex separation, respectively. The width of a layout gives a lower bound on the memory required to store the set of active mesh vertices in the stream at any time, while span is an upper bound on the time any vertex stays active; it also bounds the number of bits needed for relative vertex indexing. Both span and width influence mesh streamability. Profile, workbound, and bandwidth all affect the cost of Cholesky factorization [3], which becomes necessary when large sparse linear systems for the implicit mesh filtering problem need to be solved [13, 14].

The optimization problems associated with most of the criteria mentioned so far are NP-hard [1]. Polynomial-time approximation algorithms exist for many layout problems [1], e.g., the $O(n^{2.2})$ algorithm [20] for MinLA, but the time complexity prevents their use on large graphs. In practice, one resorts to efficient heuristic solutions. One of the best known heuristics for minimizing span (bandwidth) is the Cuthill-Mckee (CM) scheme [21]. The reverse CM (RCM) scheme [3] reverses the final order obtained with CM; compared with CM, this always reduces the profile while keeping the bandwidth unchanged. Both CM and RCM essentially conduct a degree-oriented breadth-first search. For the other costs we have mentioned so far, spectral sequencing using the Fiedler vector has been quite successful empirically. Let A be the adjacency matrix of a weighted graph and D a diagonal matrix of A's row sums, then the Fiedler vector $\mathbf{e}$, i.e., the eigenvector corresponding to the first non-zero eigenvalue of the graph Laplacian $L = D - A$ is known to minimize

$$\sum_{(u,v)\in E} A_{uv}[(\mathbf{e}_u - \mathbf{e}_v)]^2, \text{where } \mathbf{e}_i \text{ is the } i\text{-th entry of } \mathbf{e} \text{ and } \sum_{i\in V} \mathbf{e}_i = 0.$$

By sorting the entries of $\mathbf{e}$, the corresponding vertex ordering is obtained. The above quantity can be seen as a continuous relaxation of the min-2-sum for graphs and so far this has been the main motivation for using the Fiedler vector as a heuristic for computing optimal graph layouts [2, 8].

3 Kernel principal component analysis: a review

To make the paper self-complete, we brief in this section the Kernel PCA, which is utilized by our algorithm to derive sequences.

Given a set of points $\mathcal{X} = \{\mathbf{x}_i \in \mathbb{R}^d, i = 1, \ldots, n\}$, principal component analysis (PCA) finds the orthogonal major components of $\mathcal{X}$ as the eigenvectors of the covariance matrix $C_{\mathcal{X}} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}}_i)(\mathbf{x}_i - \bar{\mathbf{x}}_i)^T$, where $\bar{\mathbf{x}} = \sum_{i=1}^{n}\mathbf{x}_i$. The first principal component accounts for the direction along which the projections of $\mathcal{X}$ have the largest variation, and each subsequent component accounts for an orthogonal direction along which as much of the remaining variation as possible

lies. As a convention, we say an eigenvector is "larger" when its corresponding eigenvalue is larger.

Instead of working on $\mathcal{X}$ directly, Kernel PCA (KPCA) [22] first applies a mapping $\phi : \mathbb{R}^d \to \mathcal{F}, \mathbf{x} \mapsto \phi(\mathbf{x})$ to transform $\mathcal{X}$ to $\phi(\mathcal{X}) = \{\phi(\mathbf{x}_i)\}$ and carry out PCA on $\phi(\mathcal{X})$, where $\mathcal{F}$ is called the *feature space* that may have a much higher, and possibly infinite, dimensionality. The mapping ϕ allows for an "unfolding" of the non-linear structures of $\mathcal{X}$ into linear ones in $\mathcal{F}$, where a linear algorithm can be effective. Since the feature space $\mathcal{F}$ could have infinite dimensionality, ϕ is never explicitly defined, but implicitly defined by a kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$.

Denote by $\xi_1, \xi_2, \ldots, \xi_n$ the major components of $\phi(\mathcal{X})$ and let K be the kernel matrix, with $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and $(\lambda_i, \mathbf{U}_i)$ its i^{th} largest eigenvalue-eigenvector pair. It is shown [22] that the projection of $\phi(\mathbf{x}_i)$ onto the component ξ_m is

$$\mathbf{y}_i^m = \langle \xi_m, \phi(\mathbf{x}_i) \rangle = \frac{1}{\sqrt{\lambda_m}} \sum_{j=1}^{n} \mathbf{U}_m^j K(i,j). \tag{1}$$

Note that the discussion so far assumes that points in $\phi(\mathcal{X})$ are centered; otherwise the points have to be centered first, indirectly by transforming K through the following equation

$$K := (I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)K(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T), \tag{2}$$

where $\mathbf{1}$ is the column vector of 1's. We will show in Section 5.2 how our algorithm can bypass this step for efficient computation of mesh layouts.

If the relation between data, in our case the graph vertices, can be modeled by a kernel matrix, Kernel PCA provides a way to embed the data into a high dimensional feature space. By finding an appropriate 1-D projection of the embeddings, we can derive a linear mesh layout that has desirable properties.

4 Spectral sequencing for graph layout

In this section, we present our spectral sequencing algorithm for graph layouts, where an eigenvector of a kernel [1] matrix is used for labeling. Although we focus on mesh layouts, our discussions in this section are cast in the general context of graph layouts. To produce a sequence of mesh vertices or faces, the graph of interest can be set to the primal graph or the dual graph of the mesh, respectively. For the mesh layout problem we currently consider, edge weights will be assumed to be unit. Our approach however can trivially adapt to weighted graphs.

As mentioned before, our algorithm is not targeted to minimizing any specific layout cost; instead, it is designed to preserve the global coherence and locality information of the graph. In other words, we stipulate that to minimize the

[1] Since the entries of the kernel matrix also measure the affinities between data, we might use the terminology affinity matrix and kernel matrix interchangeably.

various layout costs, such as width, span, profile, workbound, etc., it is desirable for close-by graph vertices, based on connectivity and edge weights, to be close in the sequence. Take the *min-1-sum* problem for example. If all the adjacent vertices in the graph were consecutive in the sequence, the closest possible in 1-D, an optimal layout would be obtained.

This is of course not possible in 1-D in general, but we can look for an embedding ϕ of the graph vertices in a high dimensional space, in which adjacent vertices are always closer to each other. To obtain the final sequence, the vertices are projected onto a vector, with the condition that their mutual distances are preserved as mush as possible. Given a graph $G =< V, E >, |V| = n$ with unit weights on edges, the following is the overview of our algorithm:

1. Calculate the graph distance, $g(v_i, v_j)$, between each pair of vertices.
2. Compute an embedding $\phi(V) = \{\phi(v_i)\}$ of V, such that given any vertex v, $\forall(i,j), ||\phi(v) - \phi(v_i)||^2 < ||\phi(v) - \phi(v_j)||^2$, if and only if $g(v, v_i) < g(v, v_j)$.
3. Project points in $\phi(V)$ onto a vector $\mathbf{p}^*$, along which their relative positions are preserved as much as possible.
4. Sort the projections of $\phi(V)$ on $\mathbf{p}^*$ to obtain the resulting sequencing of the vertices.

If the projections of $\phi(V)$ onto $\mathbf{p}^*$ were distortion-free, meaning that the order among distances between vertices is totally preserved, the optimal layout in our setting would result. Note that, for the sake of clarity, we ignore several practical issues for the time being; they will be addressed in detail in Section 5.

4.1 Embedding

To find the embedding $\phi(V)$, with $\phi : V \to \mathcal{F}, v \mapsto \phi(v)$, we resort to the concepts from Kernel PCA briefed in Section 3. The first step is therefore to construct the kernel (affinity) matrix K which implicitly defines ϕ. To this end, we start by building a distance matrix W, where $W_{ij} = g(v_i, v_j)$ is the graph distance between v_i and v_j.

Once W is computed, the next step is to convert it into the kernel matrix K using a certain kernel function. One of the most popular kernels is the Gaussian radial basis function,

$$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\delta^2}}, \tag{3}$$

where δ is the kernel width. Then we have $K_{ij} = k(v_i, v_j) = \langle \phi(v_i), \phi(v_j) \rangle = e^{-\frac{W_{ij}^2}{2\delta^2}}$. Note that $k(v_i, v_i)$, the kernel between any vertex and itself, is equal to 1, due to the fact that $g(v_i, v_i) = 0$.

Now let's examine the Euclidean distances among the vertices in the feature space $\mathcal{F}$. Given any two vertices v_i, v_j, denote by $\mathcal{W}_{ij}$ their distance in $\mathcal{F}$, we

have:

$$\begin{aligned}\mathcal{W}_{ij}^2 &= ||\phi(v_i) - \phi(v_j)||^2 \\ &= (\phi(v_i) - \phi(v_j))^T (\phi(v_i) - \phi(v_j)) \\ &= \phi(v_i)^T \phi(v_i) + \phi(v_j)^T \phi(v_j) - 2\phi(v_i)^T \phi(v_j) \\ &= 2 - 2\phi(v_i)^T \phi(v_j) \\ &= 2 - 2k(v_i, v_j) \\ &= 2 - 2e^{-\frac{W_{ij}^2}{2\delta^2}}\end{aligned}$$

Thus distances in $\mathcal{F}$ are seen to be proportional to those in the graph, in that the order among distances between vertices in the graph is preserved in the feature space. Specifically, since the graph edges have unit length, neighboring vertices in the graph are always closest to each other in the feature space $\mathcal{F}$. We refer to this property as distance monotonicity.

Before we continue, we point out that the points in $\phi(V)$ are not centered when the kernel matrix K is constructed the way described in this section. In order for the algorithm in the following section to apply, K should be processed using equation (2) immediately after it is constructed. As a result, $\phi(V)$ is now centered.

4.2 Sequencing

At this point, the vertex embedding $\phi(V)$ is already known implicitly through K. It is also true that the distances between pairs of vertices in the graph are relatively preserved in the feature space $\mathcal{F}$, in the sense that the close-by vertices in the graph are also close in the feature space. Therefore, the positioning of the vertices in $\phi(V)$ provides a good start to extract a vertex sequence to minimize the various layout costs we have defined so far.

We compute a vertex sequence by projecting each $\phi(v) \in \phi(V)$ onto a vector $\mathbf{p}$. Afterwards, the projections can be simply sorted to obtain the sequence. Since after the projection the dimensionality of the embedding space for vertices is reduced to 1, to obtain a smaller layout cost for the resulting sequence, it is desirable to preserve their mutual distances, hence the relative positions, as much as possible. To this end, we choose the optimal direction $\mathbf{p}^*$ subject to the following objective function

$$\mathbf{p}^* = \underset{\mathbf{p} \in \mathbb{R}^{\dim(\mathcal{F})}, ||\mathbf{p}||=1}{\operatorname{argmax}} \sum_{i<j} ||\mathbf{p}^T(\phi(v_i) - \phi(v_j))||^2. \tag{4}$$

The underlying reasoning is quite simple. Since the length of a vector projected onto $\mathbf{p}$ is always smaller than the length the original vector, collectively maximizing the sum of projected distances tends to preserve the original distances.

In equation(4),

$$\begin{aligned}
&\sum_{i<j} ||\mathbf{p}^T(\phi(v_i) - \phi(v_j))||^2 \\
&= \sum_{i<j} \mathbf{p}^T \left(\phi(v_i) - \phi(v_j)\right) \left(\phi(v_i) - \phi(v_j)\right)^T \mathbf{p} \\
&= \mathbf{p}^T \left((n-1) \sum_{i=1}^{n} \phi(v_i)\phi(v_i)^T \right) \mathbf{p} - \mathbf{p}^T \left(\sum_{i \neq j} \phi(v_i)\phi(v_j)^T \right) \mathbf{p} \\
&= \mathbf{p}^T \left((n-1) C_{\phi(V)} \right) \mathbf{p} - \mathbf{p}^T \left(\sum_{i \neq j} \phi(v_i)\phi(v_j)^T \right) \mathbf{p} \,,
\end{aligned} \tag{5}$$

where $C_{\phi(V)} = \sum_{i=1}^{n} \phi(v_i)\phi(v_i)^T$ is the covariance matrix of the point set $\phi(V)$. Since $\phi(V)$ is centered, $\sum_{j \neq i} \phi(v_j)^T = -\phi(v_i)^T$. As a result,

$$\sum_{i \neq j} \phi(v_i)\phi(v_j)^T = \sum_{i} \phi(v_i) \sum_{j \neq i} \phi(v_j)^T = \sum_{i} \phi(v_i)(-\phi(v_i))^T = -C_{\phi(V)}.$$

Substitute this into equation (5), we have

$$\sum_{i<j} ||\mathbf{p}^T(\phi(v_i) - \phi(v_j))||^2 = n\mathbf{p}^T C_{\phi(V)} \mathbf{p}.$$

Consequently, equation (4) is maximized when $\mathbf{p}$ is the largest eigenvector of $C_{\phi(V)}$. Namely, $\mathbf{p}^*$ should be taken as the first principal component of the point set $\phi(V)$.

If we denote the projections of the points in $\phi(V)$ onto $\mathbf{p}^*$ as a vector $\phi(V)_{\mathbf{p}^*}$, the question is how to compute $\phi(V)_{\mathbf{p}^*}$ without knowing ϕ and $\mathbf{p}^*$ explicitly. Referring to equation (1), we see that this problem can be readily solved since to compute the projection of embedded points to the major components in the feature space, all we need are the kernel matrix K and its eigenvalues and eigenvectors. For $K = U\Lambda U^T$, we have $U = [\mathbf{U}_1|\mathbf{U}_2|, \ldots, |\mathbf{U}_n]$, $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, where $\lambda_1 \geq \lambda_2 \geq, \ldots, \geq \lambda_n$ are the eigenvalues. Denote $Y = [\mathbf{Y}_1|\mathbf{Y}_2|, \ldots, |\mathbf{Y}_n] = [\mathbf{y}_1|\mathbf{y}_2|, \ldots, |\mathbf{y}_n]^T$, the matrix form of equation (1) is

$$Y = KU\Lambda^{-\frac{1}{2}}. \tag{6}$$

Thus, $\mathbf{y}_i$ is the projection of $\phi(v_i)$ onto the principal components of $\phi(V)$, while $\mathbf{Y}_i$ contains the projections of all $\phi(v)$'s onto the i^{th} principal component of $\phi(V)$. In particular, $\mathbf{Y}_1$ is the projections of $\phi(V)$ onto the first principal component, hence the projection $\phi(V)_{\mathbf{p}^*}$ that we want to compute. From (6), we also see that $Y = U\Lambda\Lambda^{-\frac{1}{2}} = U\Lambda^{\frac{1}{2}}$. As what we are interested in is the vertex ordering given by $\mathbf{Y}_1$ instead of its absolute value, we can simply set $\mathbf{Y}_1 := \mathbf{U}_1$, ignoring the scaling factor $\sqrt{\lambda_1}$, for the sole purpose of ordering.

To summarize, the spectral sequencing algorithm itself, without dealing with certain practical issues, turns out to be quite straightforward, though the underlying theory requires non-trivial derivations. It simply constructs the centered kernel (affinity) matrix K and computes its eigenvalue decomposition $K = U\Lambda U^T$. Then the entries of the largest eigenvector $\mathbf{U}_1$ are sorted to produce the sequence for V, answering the graph layout problem.

5 Some practical issues

5.1 Choice of kernel function and kernel width

To convert a distance matrix to a kernel matrix, we use the Gaussian kernel, given in equation (3). Note that other kernels, e.g., step function, exponential, polynomial and rational polynomial kernels, are also possible. In fact, the use of the Fiedler vector can be seen as a special case of our general paradigm, when applied to the dual mesh graph. Specifically, the dual graph of a triangle mesh is 3-regular. Thus the eigenvectors of the graph Laplacian $L = D - A$ coincide with the eigenvectors of A, which is an affinity matrix with a step function kernel of width 1. We have experimented with other kernels and have not found particular reasons to prefer one over the other, but this issue remains to be investigated in our future work.

Another issue is the selection of the kernel width δ. In the context of spectral clustering [23, 24], selecting an appropriate kernel width is not an easy task. The value of δ influences the quality of the clustering result dramatically, since the value of δ sets the threshold for considering points as from the same group or not. However, this is not a problem for us because what is essentially decisive in our application is the relative positioning of the embeddings in $\phi(V)$, and this is determined by the distance monotonicity property independent of δ, as discussed in Section 4.1.

We shall point out that the invariance of δ to the layout results is only true when the kernel matrix K is centered. Practically, the K we use is not centered, in which case, δ should be sufficiently large. This will be discussed in the next section.

5.2 Avoiding centering kernel matrix K

Till now, the point set $\phi(V)$ has to be centered for our approach to apply. This is done indirectly by pre-processing the kernel matrix K with equation (2). One major drawback of centering K is its inefficiency when the size of K is large. As we shall describe in Section 5.4, the full K is not available for centering due to the subsampling, which is necessary for handling large data sets. Practically, we therefore do not have the centered K and its dominant eigenvector $\mathbf{U}_1$ to derive the sequence. Alternatively, we propose to use the *subdominant* eigenvector $\mathbf{U}_2$ of the uncentered K, which can be shown to be a close approximation to the dominant eigenvector, $\bar{\mathbf{U}}_1$, of the centered K, denoted by $\bar{K}$ in what follows.

Since $K_{ii} = \phi(v_i)^T \phi(v_i) = 1$, it is clear that each $\phi(v_i)$ has unit length. Then K_{ij} is simply the cosine of the angle between vectors $\phi(v_i)$ and $\phi(v_j)$. Given any vector $\phi(v_i)$, it can be concluded that $\forall j \neq i, \phi(v_j)$ sits in a cone rooted at the origin, along vector $\phi(v_i)$ and with a half angle θ_i, where θ_i is determined by the smallest entry K_{it} of the i^{th} row of K. If the kernel width δ is set to a sufficiently large value, K_{it} will also be close to 1 and, accordingly, θ_i will be very small. This situation is depicted in Figure 1. Each point in $\phi(V)$ is associated with a cone and the *intersection* of these cones gives a region, R, in which all the points lie. In the figure, R is indicated by the dotted closed curve. Note that the cone and the region R are enlarged for illustration purpose. Note that if $\phi(V)$ is centered, the region R will move to the place indicated by $\bar{R}$. Roughly speaking, the idea is to show that the major components of the points in $\bar{R}$ are approximately the same as the major components of the points in R, only shifted by one position.

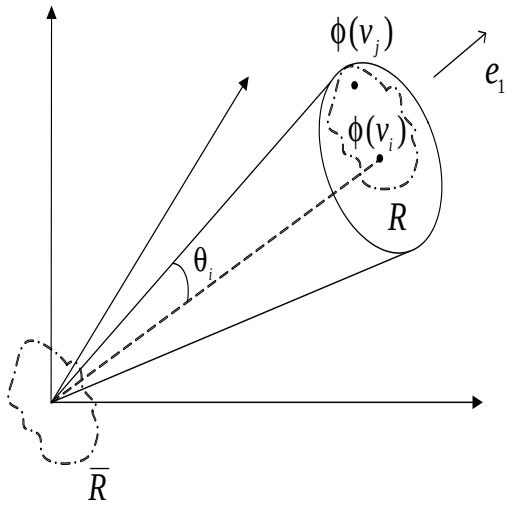

Fig. 1. Distribution of embedded points in the feature space.

Denote by $\Phi = [\phi(v_1)|\phi(v_2)|, \ldots, |\phi(v_n)]$, then the covariance matrix of $\phi(V)$ is given by $C_{\phi(V)} = \Phi\Phi^T$. The largest eigenvector $\mathbf{e}_1$ of $C_{\phi(V)}$ is the vector $\mathbf{p}$ which maximizes the variance $\sum_i ||\mathbf{p}^T \phi(v_i)||$. Since all $\phi(v)$'s have unit length and small angles between each other, the region R is small and the distance between any point inside R and the origin is about 1. As a result, $\mathbf{e}_1$ should roughly point from the origin to the center of R to maximize the variance, as illustrated in Figure 1. With the same reasoning, R is "shallow" along the direction of $\mathbf{e}_1$. Consequently, the space spanned by $\phi(V)$ is close to the null space of $\mathbf{e}_1$. The subsequent eigenvectors $\{\mathbf{e}_2, \mathbf{e}_3, \ldots\}$ of $C_{\phi(V)}$, in the null space of $\mathbf{e}_1$, are then a good approximation to the principal components which characterize the intrinsic distribution of $\phi(V)$. On the other hand, if $\phi(V)$ is centered to

$\bar{\phi}(V)$, R moves to $\bar{R}$. We denote the corresponding centered covariance matrix by $C_{\bar{\phi}(V)} = \bar{\Phi}\bar{\Phi}^T$, where $\bar{\Phi} = [\bar{\phi}(v_1)|\bar{\phi}(v_2)|, \ldots, |\bar{\phi}(v_n)]$ and let $\{\bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2, \ldots\}$ be the leading eigenvectors of $C_{\bar{\phi}(V)}$. Since $\phi(V)$ and $\bar{\phi}(V)$ have the same intrinsic distribution and are displaced only by a translation, they should have the same *intrinsic* major components, namely we have $\bar{\mathbf{e}}_1 \approx \mathbf{e}_2$, $\bar{\mathbf{e}}_2 \approx \mathbf{e}_3$, etc., where the approximation is exact if the space spanned by $\phi(V)$ coincides with the null space of $\mathbf{e}_1$.

Finally, we are prepared to show that $\mathbf{U}_2 \approx \bar{\mathbf{U}}_1$. Since $\bar{K} = \bar{\Phi}^T\bar{\Phi}$, it is easy to prove that $\bar{K}$ and $C_{\bar{\phi}(V)}$ have the same eigenvalues and $\bar{\Phi}^T\bar{\mathbf{e}}_1$ is the largest eigenvector of $\bar{K}$, i.e. $\bar{\mathbf{U}}_1 = \bar{\Phi}^T\bar{\mathbf{e}}_1$. Similarly, $\mathbf{U}_2 = \Phi^T\mathbf{e}_2$. Therefore, we only need to show that $\bar{\Phi}^T\bar{\mathbf{e}}_1 \approx \Phi^T\mathbf{e}_2$. Let us write $\psi = \frac{1}{n}\sum_i \phi(v_i)$, then we have $\Phi^T = \bar{\Phi}^T + \mathbf{1}\psi^T$. Therefore $\Phi^T\mathbf{e}_2 = (\bar{\Phi}^T + \mathbf{1}\psi^T)\mathbf{e}_2 = \bar{\Phi}^T\mathbf{e}_2 + \mathbf{1}\psi^T\mathbf{e}_2 \approx \bar{\Phi}^T\bar{\mathbf{e}}_1 + \mathbf{0} = \bar{\Phi}^T\bar{\mathbf{e}}_1$. Recall that $\bar{\mathbf{e}}_1 \approx \mathbf{e}_2$ and $\psi^T\mathbf{e}_2 \approx \mathbf{0}$ since $\psi \approx \mathbf{e}_1$.

Figure 2 provides an experimental comparison between $\mathbf{U}_2$ and $\bar{\mathbf{U}}_1$. The kernel matrices K and $\bar{K}$ are computed from the dual graph of a mesh with 600 faces. The kernel function in equation (3) is used. As we can see from the plots, the approximation quality is quite good when δ is sufficiently large. In our experiments in Section 6, we set δ to be the average of the distances computed.

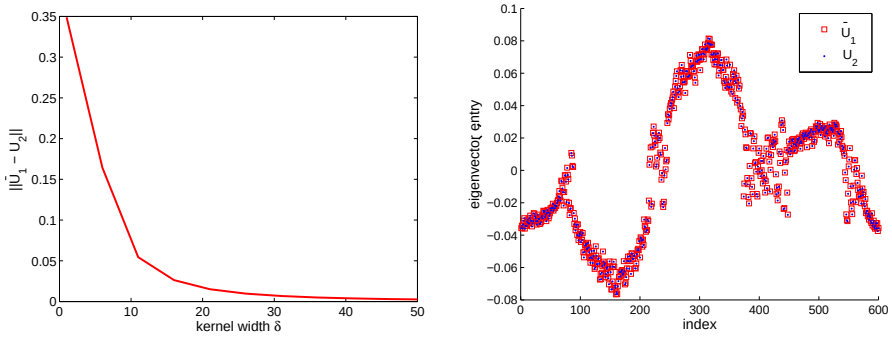

Fig. 2. Influence of δ to the approximation quality of $\mathbf{U}_2$ to $\bar{\mathbf{U}}_1$. The left plot shows that $||\bar{\mathbf{U}}_1 - \mathbf{U}_2||$ decreases quickly while δ increases. The right plots the two eigenvectors when $\delta = 50$, in which case the dominant (subdominant) eigenvalue of $\bar{K}$ (K) is $\bar{\lambda}_1 = 24.5022$ ($\lambda_2 = 24.4998$), and $||\bar{\mathbf{U}}_1 - \mathbf{U}_2|| = 0.0025$.

5.3 Positive SemiDefiniteness of K

It is known in Kernel PCA literature that a valid kernel matrix has to be positive semi-definite. However, K constructed so far is generally not positive semi-definite. To overcome this problem, K can be preprocessed to $\tilde{K}$ which is positive semi-definite. Suppose $K = U\Lambda U^T$ and has r non-negative eigenvalues

$(\lambda_1, \ldots, \lambda_r)$, we can set $\tilde{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_r, 0, \ldots)$; then the kernel matrix for use is reconstructed as $\tilde{K} := U\tilde{\Lambda}U^T$.

As a matter of fact, our experiments show that the results produced by using either K or $\tilde{K}$ are almost the same. This is because albeit negative eigenvalues exist, their number and especially magnitude are much smaller than those of leading positive eigenvalues. This suggests that K and $\tilde{K}$ are quite similar and the corresponding inferred positions of points in $\phi(V)$ are basically the same. Therefore we simply use K for the sake of efficiency.

5.4 Subsampling

The algorithm so far is not able to handle large meshes, unless we lift the two bottlenecks in computing the 1-D embedding $\mathbf{U}_2$. The first one is to compute the distance matrix W. When Dijkstra's algorithm is used, the complexity of computing W is $O(n^2 \log n)$. The second one is to calculate the eigenvalue decomposition of K, of which the complexity is $O(n^3)$. Although it is possible to reduce the complexity of this step by computing only leading eigenvectors using software packages such as ARPACK [25], the overhead is still unbearable because K is not sparse, especially when n is large. To overcome these two difficulties, we use subsampling and the Nyström method [26].

By means of subsampling, not all pair-wise graph distances are needed; instead, a subset of vertices are carefully selected, and only the graph distances between these sampled vertices and the remaining vertices are computed. Thus we only need to construct a partial W, and the complexity therein is reduced to $O(mn \log n)$, where m is the number of samples. We can then write the resulting kernel matrix in block form

$$K = \begin{bmatrix} P & Q \\ Q^T & S \end{bmatrix}.$$

Without loss of generality, assume $P \in \mathbb{R}^{m \times m}$ encodes the kernels within sampled vertices and $Q \in \mathbb{R}^{m \times (n-m)}$ contains kernels between sampled and unsampled vertices. Due to sampling, only the sub-block $[P\ Q]$ is known. The question that remains is how to find the second largest eigenvector $\mathbf{U}_2$ of K by only knowing $[P\ Q]$. Nyström method [26] serves this purpose. Let $P = F\Sigma F^T$ and $K = U\Lambda U^T$, Nyström method approximates the m leading eigenvectors of K as

$$U_{(1,\ldots,m)} := \begin{bmatrix} \Sigma \\ Q^T \Sigma \Lambda^{-1} \end{bmatrix}. \tag{7}$$

Note that the eigenvectors extrapolated through Nyström method are the leading eigenvectors and only m of them can be found. With equation (7), we are able to find an approximation to $\mathbf{U}_2$ and the complexity is $O(mn \log n) + O(m^3)$. As we will see later, our algorithm applies aggressive sampling, using a fixed number ($m = 10$) of samples for meshes with tens of thousands of vertices. Since $m << n$, the overall complexity becomes $O(mn \log n)$, allowing us to process large meshes efficiently. Another practical issue is the selection of the small sample set without sacrificing too much quality. Based on our previous work [26],

we resort to farthest point sampling, which chooses samples that are mutually furthest away. Our experiments demonstrate that Nyström approximation with furthest point sampling works remarkably well for the layout problem, at an extremely low sampling rate.

6 Experimental results

This section presents an experimental comparison which evaluates the quality of the layout generated by the Fiedler vector, the Cuthill-Mckee scheme and our approach, referred to as Laplacian, CM, and Affinity, respectively. Given a mesh, its vertex or face sequence is generated by considering the primal or dual graph, respectively. In our experiments, we consider six quality measures of the layout: span, width, profile, workbound, 1-discrepancy and 2-discrepancy, where the first two are of particular interest to mesh streaming [15].

Figure 3 shows the triangle meshes used in our experiments, which are performed on a Pentium 1.7GHz processor with 1GB RAM. Note that models with boundary (Crater), elongated aspect ratio (Isis) and non-zero genus (Rocker Arm) are tested on. Table 1 presents the characteristics of these models in conjunction with the time required to compute the vertex and face sequence by our algorithm. Note that CM works the fastest among the three, while Affinity and Laplacian perform similarly in speed.

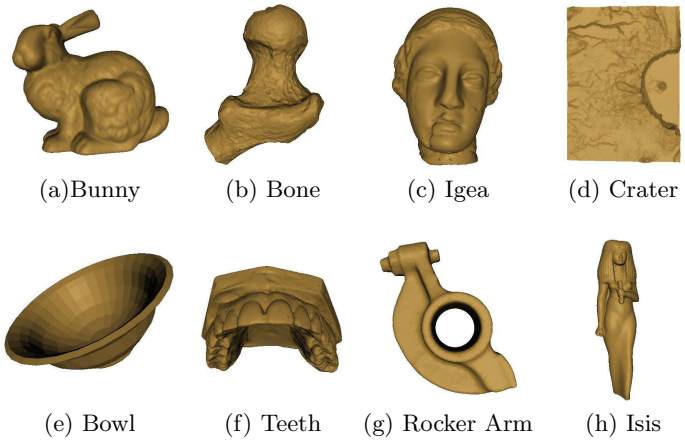

Fig. 3. Models used in the experiments.

Figures 4 and 5 show the comparison for the six layout measures, when considering the mesh primal graph and the dual graph, respectively. It can be seen that the CM obtains the best results in terms of span, while the Laplacian operator provides the best results in terms of the other measures. However,

Table 1. Characteristics of models and timing in seconds, I/O excluded.

Model	**vertex #**	**face #**	**time (Vertex)**	**time (Face)**
Bunny	34,834	69,451	2.00	3.27
Bone	50,002	100,000	3.14	5.00
Igea	60,002	120,000	4.20	7.14
Crater	100,000	199,114	7.31	11.87
Bowl	102,402	204,800	7.45	12.21
Teeth	100,002	200,000	7.58	13.58
Rocker Arm	160,704	321,408	15.50	23.97
Isis	187,644	375,276	23.80	35.83

since the Affinity outperforms the Laplacian in span and the CM in width and other measures, it provides a trade-off between these two sets of measures, thus providing potential benefits to applications where all these measures influence performance simultaneously. Since the models we use have different structural properties, we believe that our result is model independent.

It is worth pointing out that Affinity tends to outperform both CM and Laplacian in terms of span and width on dual graphs (for face sequencing). This can be seen from the span measure in Figure 5, where the Affinity presents the best results for some models. We believe this has something to do with the regularity of the graphs. Note that since the tested models are triangle meshes, the dual graphs are regular graphs. This issue is currently under investigation.

Figure 6 illustrates another test to verify that the subdominant eigenvector $\mathbf{U}_2$ of the uncentered kernel matrix is a good approximation to the dominant eigenvector $\bar{\mathbf{U}}_1$ of the centered kernel matrix, by comparing the span and width measures against different kernel width. It is clear that using $\mathbf{U}_2$ or $\bar{\mathbf{U}}_1$ produces similar results when a sufficiently large kernel width δ is used. In order to obtain $\bar{\mathbf{U}}_1$ from the centered kernel matrix, subsampling is not applied (Section 5.2). As seen from the plots, $\bar{\mathbf{U}}_1$ and $\mathbf{U}_2$ produce almost the same result after δ is sufficiently large. After the Affinity enters the stable region, its performance lies between Laplacian and CM consistently.

In our experiments, we fix the number of samples to 10, which is extremely small against the size of meshes used in our experiments. To demonstrate the effectiveness of subsampling, we provide results in Table 2 for span and width obtained from various models with and without sampling. In this test, the mesh primal graph is considered. As can be observed from the table, sampling is able to produce results comparable to those obtained when sampling is not applied. Surprisingly, it is sometimes able to produce even better results. Note that we only consider relatively small-sized models because the computational overhead without subsampling prevents the use of large models.

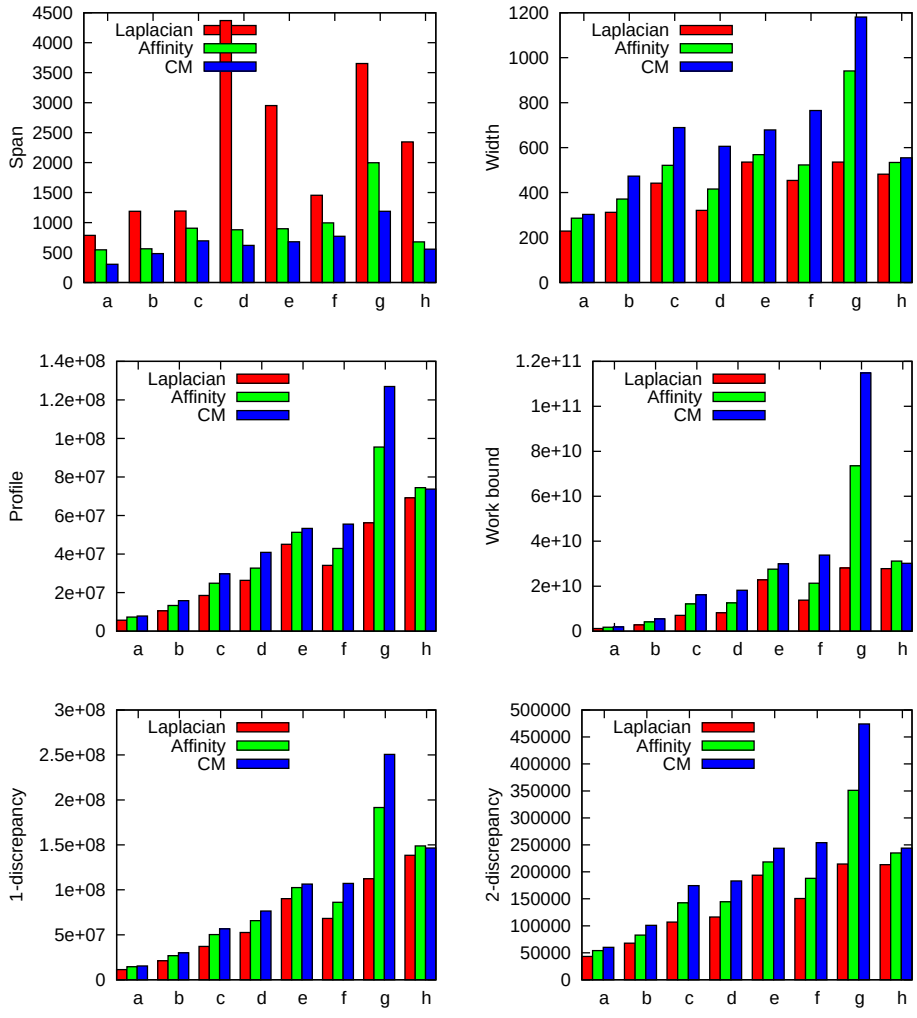

Fig. 4. Comparison of layout quality measures for primal graph (vertex) sequencing of different models: (a) Bunny (b) Bone, (c) Igea, (d) Crater, (e) Bowl, (f) Teeth, (g) Rocker Arm, (h) Isis.

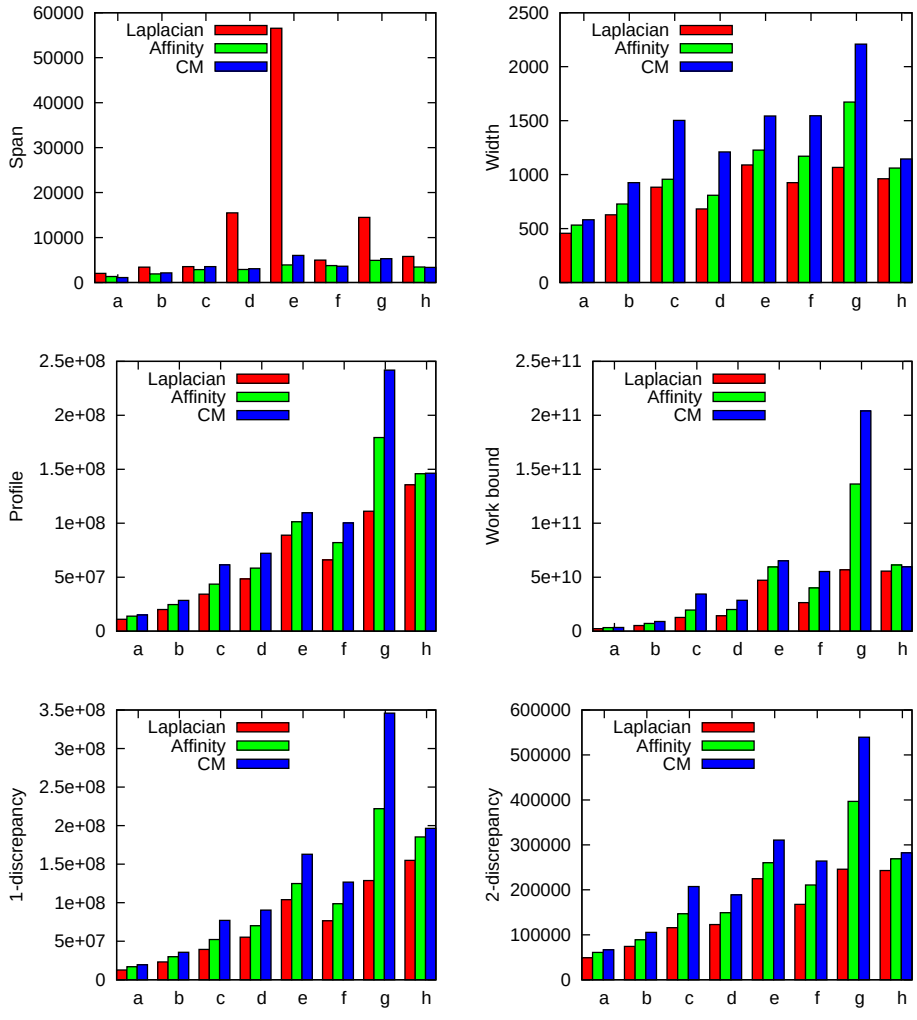

Fig. 5. Comparison of layout quality measures for dual graph (face) sequencing of different models: (a) Bunny (b) Bone, (c) Igea, (d) Crater, (e) Bowl, (f) Teeth, (g) Rocker Arm, (h) Isis.

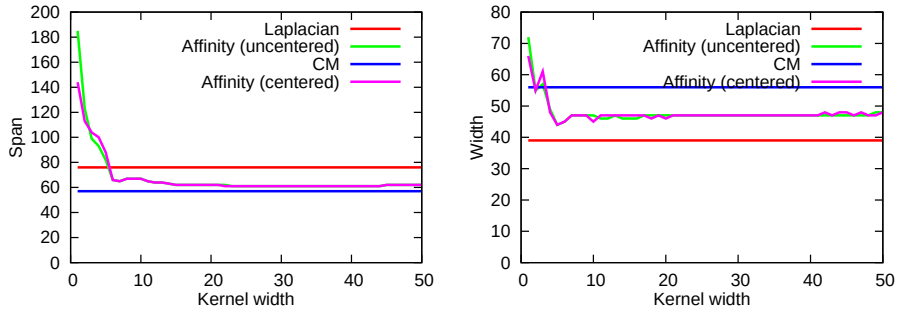

Fig. 6. Comparison of layout quality measures for an simplified Isis model. The Affinity (uncentered) and Affinity (centered) legends correspond to our algorithm making use of $\bar{\mathbf{U}}_1$ and $\mathbf{U}_2$, respectively.

Table 2. Comparison of span and width for various mesh graphs when subsampling is either applied or not.

Vertex #	**Without sampling**		**With sampling**	
	Span	Width	Span	Width
865	89	61	87	62
1002	164	48	121	41
1002	108	70	116	71
1002	125	58	142	61
1032	126	83	128	66
1121	131	67	149	72
1210	160	70	178	74
1872	210	92	186	82
2360	179	97	169	100

7 Conclusion and future work

In this paper, we present a spectral sequencing algorithm for linear mesh layouts. We study this problem in the more general context of linear graph layouts. Independent of any specific cost measure, we abstract that a desirable vertex sequence should be one in which close-by vertices in the graph are also close-by in the sequence. To this end, Kernel PCA is utilized to embed graph vertices in a high dimensional feature space so as to preserve their relative distances in the input graph. A sequence, the layout, is then obtained by projecting the embeddings onto a vector, so that the mutual distances between the embeddings are least distorted according to an appropriately defined objective function.

To overcome the computational overhead required by kernel matrix generation and eigenvalue decomposition, we resort to subsampling and eigenvector extrapolation, using the Nyström method, and propose to use the subdominant

eigenvector of the uncentered kernel matrix, rather than the dominant eigenvector of the centered kernel matrix. An intuitive argument is provided to validate our approach, assuming that a Gaussian kernel with a sufficiently large kernel width is applied in Kernel PCA.

Our extensive experiments demonstrate that for span and width, the two principal quality measures for mesh streaming, as well as other graph layout cost criteria, e.g., profile and workbound, our algorithm potentially provides a better trade-off compared to ordering schemes based on localized graph traversal, e.g., Cuthill-Mckee, and spectral sequencing using the Fiedler vector.

The geometric appeal of our framework and analysis based on Kernel PCA has shed some light on possible further improvement of layout qualities. One particularly intriguing problem is to investigate the distribution properties of the vertex embedding in the feature space. It seems desirable to embed the vertices in a way such that after the projection, less "overlaps" of distant embeddings would occur. It is therefore important to have most of the variance along the first principal component. We believe this is related to the "gap" between the first and second eigenvalues of the centered kernel matrix. To this end, we would like to investigate the influence of locally adaptive kernel functions, e.g., as suggested in [24], and more sophisticated distance measures between graph vertices that take into consideration more global mesh connectivity information.

References

1. Díaz, J., Petit, J., Serna, M.: A survey of graph layout problems. ACM Computing Survey **34**(3) (2002) 313–356
2. Barnard, S.T., Pothen, A., Simon, H.D.: A spectral algorithm for envelope reduction of sparse matrices. In: Proc. ACM/IEEE Conference on Supercomputing. (1993) 493–502
3. George, A., Liu, W.H.: Computer Solution of Large Sparse Positive Definite Systems. Prentice-Hall (1981)
4. Corso, G.M.D., Romani, F.: Heuristic spectral techniques for the reduction of bandwidth and work-bound of sparse matrices. Technical report, Universitá di Pisa, Dipartimento di Informatica (2001)
5. Gibbs, N., Poole, W., Stockmeyer, P.: An algorithm for reducing the bandwidth and profile of a sparse matrix. SIAM J. on Numerical Analysis **13** (1976) 236–249
6. Hur, S.W., Willis, J.: Relaxation and clustering in a local search framework: application to linear placement. In: Proc. ACM/IEEE Conference on Design Automation. (1999) 360–366
7. Karp, R.M.: Mapping the genome: some combinatorial problems arising in molecular biology. In: Proc. ACM Symposium on Theory of Computing. (1993) 278–285
8. Mohar, B., Poljak, S.: Eigenvalues in combinatorial optimization. In Brualdi, R.A., Friedland, S., Klee, V., eds.: IMA Volumes in Mathematics and Its Applications. Volume 50. Springer-Verlag (1993) 107–151
9. Brin, S., Page, L.: Anatomy of a large-scale hypertextual web search engine. In: Proceedings of the 7th International World Wide Web Conference. (1998) 107–117
10. Yoon, S.E., Lindstrom, P., Pascucci, V., Manocha, D.: Cache-oblivious mesh layouts. ACM Trans. Graph. **24**(3) (2005) 886–893

11. Bogomjakov, A., Gotsman, C.: Universal rendering sequences for transparent vertex caching of progressive meshes. Computer Graphics Forum **21**(2) (2002) 137–148
12. Isenburg, M., Lindstrom, P., Gumhold, S., Snoeyink, J.: Large mesh simplification using processing sequences. In: Proc. of the 14th IEEE Visualization. (2003) 61–68
13. Zhang, H., Fiume, E.: Butterworth filtering and implicit fairing of irregular meshes. In: Proceedings of Pacific Graphics. (2003) 502–506
14. Kim, B.M., Rossignac, J.: Geofilter: Geometric selection of mesh filter parameters. In: Computer Graphics Forum. Volume 24. (2005) 295–302
15. Isenburg, M., Lindstrom, P.: Streaming meshes. In: IEEE Visualization. (2005)
16. Vo, H.T., Callahan, S.P., Lindstrom, P., Pascucci, V., Silva, C.T.: Stream simplification of tetrahedral meshes. (2005)
17. Koren, Y., Harel, D.: A multi-scale algorithm for the linear arrangement problem. In: WG '02: International Workshop on Graph-Theoretic Concepts in Computer Science. (2002) 296–309
18. Press, W., Tekolsky, S., Vetterling, W., Flannery, B.: Numerical Recipies in C. Cambridge Univ. Press (1992)
19. Koren, Y., Carmel, L., Harel, D.: Ace: A fast multiscale eigenvector computation for drawing huge graphs. In: IEEE Information Visualization. (2002) 137–144
20. Bar-Yehuda, R., Even, G., Feldman, J., Noar, J.: Computing an optimal orientation of a balanced decomposition tree for linear arrangement problems. Journal of Graph Algorithms and Applications **5**(4) (2001) 1–27
21. Cuthill, E., McKee, J.: Reducing the bandwidth of sparse symmetric matrices. In: Proc. 24th Nat. Conf. ACM. (1969) 157–172
22. Schölkopf, B., Smola, A., Muller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation **10** (1998) 1299–1319
23. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: analysis and an algorithm. In: NIPS. (2002) 857–864
24. Manor, L., Perona, P.: Self-tuning spectral clustering. In: NIPS. (2004)
25. Lehoucq, R., Maschhoff, K., Sorensen, D., Yang, C.: (Arpack)
26. Liu, R., Jain, V., Zhang, H.: Subsampling for efficient spectral mesh processing. In: Proceedings of Computer Graphics International 2006 (to appear). (2006)