Joint Graph Layouts for Visualizing Collections of Segmented Meshes

Jing Ren, Jens Schneider, Maks Ovsjanikov, and Peter Wonka

Abstract—We present a novel and efficient approach for computing joint graph layouts and then use it to visualize collections of segmented meshes. Our joint graph layout algorithm takes as input the adjacency matrices for a set of graphs along with partial, possibly soft, correspondences between nodes of different graphs. We then use a two stage procedure, where in the first step, we extend spectral graph drawing to include a consistency term so that a collection of graphs can be handled jointly. Our second step extends metric multi-dimensional scaling with stress majorization to the joint layout setting, while using the output of the spectral approach as initialization. Further, we discuss a user interface for exploring a collection of graphs. Finally, we show multiple example visualizations of graphs stemming from collections of segmented meshes and we present qualitative and quantitative comparisons with previous work.

Index Terms—Multi-graph layout, spectral graph layout, multi-dimensional scaling, topological exploration

1 Introduction

Graphs provide a convenient and commonly used representation of visual data, such as segmented 3D meshes, 3D scenes, and images. For example, a segmented 3D mesh can be encoded as a graph, where each node represents a component on the shape and the edges represent connectivity between the components. A 3D scene can be represented as a graph with objects as nodes and edges encoding relationships between objects.

To visually analyze a single graph one has to find an informative embedding or layout in two dimensions. Such a layout is typically driven by the idea that similar nodes should be placed close together and dissimilar nodes should be further apart. Popular algorithms for such a layout are metric multi-dimensional scaling, spectral graph drawing [1], or the algorithm of Kamada and Kawai [2], among myriad others.

For analyzing collections of graphs, however, it is no longer sufficient to provide a layout for each graph independently. A visual comparison between graphs in a collection should help answer questions such as: what components occur in one graph, but not in another? What is the dominant structure of the collection? In which parts of the graphs does the collection exhibit most variability, which graphs are outliers? Which one of the several given collections has the largest variability, etc. Answering such questions can be greatly simplified if the graphs are laid out in a consistent manner, while respecting the properties of each graph as much as possible.

In the first part of the paper (see Sec. 3), we propose two new algorithms for jointly embedding a collection of graphs in a consistent manner. The first algorithm is an extension of spectral graph drawing to multiple graphs, while the second is an extension of metric multi-dimensional scaling designed similarly to jointly handle sets of graphs. While each of these algorithms is useful by itself, the first algorithm can quickly find a good approximate solution, but has some difficulty laying out the local details. We therefore use these algorithms in sequence with the first algorithm providing the initialization for the second one. In Fig. 1 we illustrate the benefit of our joint embedding compared to embedding each graph individually. An interesting challenge that we tackle in our work is that nowhere in our pipeline do we only rely on precise node-to-node correspondences, or assume that all graphs have the same number of nodes. Instead, our framework can deal with partial, soft and group-wise correspondences. This issue is especially important for graphs stemming from segmented meshes, because these graphs often have multiple nodes with the same label (e.g. a cow typically has four nodes labeled leg). In the evaluation section (see Sec. 5), we demonstrate superior performance of our algorithm compared to the state-of-the-art in a user study. We compare to a baseline method of embedding each graph individually [2] and a state-of-the-art joint embedding method using MDS [3] with virtual edges [4].

To summarize, our main contributions are:

1) We introduce two new algorithms for computing joint graph layouts. First, we propose an extension of the classical spectral graph drawing method to the setting of multiple graphs and then we introduce an efficient and elegant majorization approach for consistently refining multiple layouts.
2) We propose and evaluate a solution to our problem formulation by comparing to other state-of-the-art methods in a user study and further we evaluate our optimization algorithm by comparing against BFGS, a classical quasi-Newton optimization algorithm.

The reason why we are particularly interested in study-

- *J. Ren, J. Schneider, and P. Wonka are with the Visual Computing Center (VCC), King Abdullah University of Science and Technology (KAUST), Thuwal, 23955, Saudi Arabia.*
- *M. Ovsjanikov is with Laboratoire d'Informatique (LIX), École Polytechnique, 91128 Palaiseau Cedex, France.*

Manuscript received May 08, 2017; revised August 14, 2017.

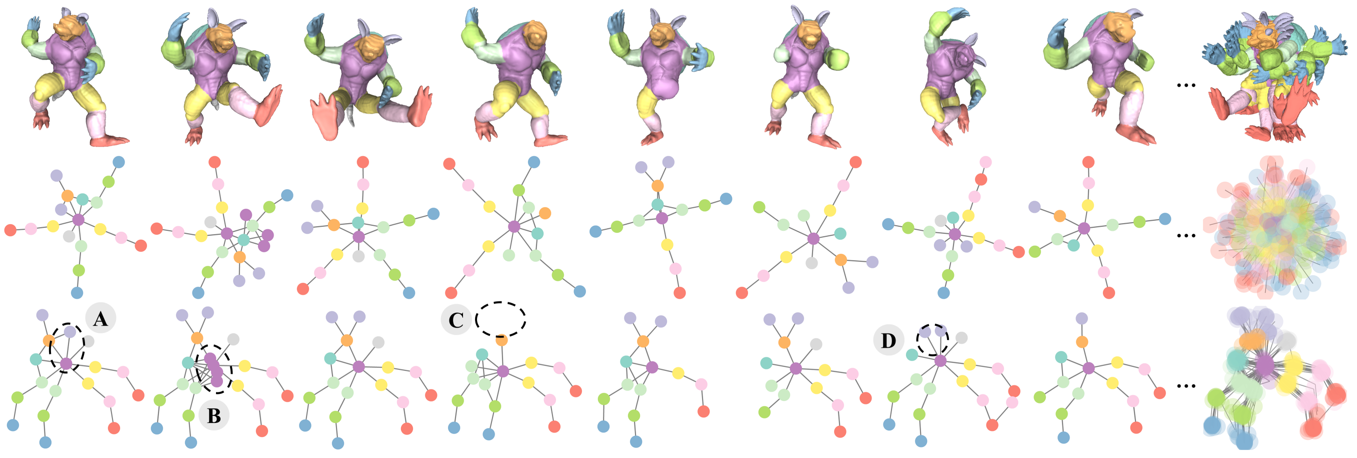


Fig. 1: First row: a collection of segmented and labeled 3D Armadillo models. Second row: the graphs corresponding to the shapes, where each node represents a connected component on a shape and edges show their adjacency. Each graph layout is computed independently using the Kamada-Kawai algorithm. Third row: graph layouts obtained with our joint embedding algorithm. The last column shows the overlaid shapes and graphs of the whole collection. We also highlight some variations in the segmentation data, e.g.: (A) one ear is wrongly connected to the torso, (B) the torso is decomposed into four separate parts, (C) two ears are missing, and (D) the head is wrongly labeled as torso.

ing graphs of segmented meshes is because segmented meshes are at the core of popular mesh processing algorithms, e.g. machine learning for mesh segmentation, component-based shape synthesis, and scene understanding for 3D scanning.

2 Related Work

Several classical graph drawing algorithms for node-link diagrams are iterative and compute forces on nodes of the layout, e.g. [2], [5], [6], or they are built on dimension reduction algorithms such as multi-dimensional scaling [3], [7], PCA, locally-linear embedding, and Laplacian Eigenmaps [8]. For a broader review of graph visualization techniques, we refer the reader to recent surveys [9]. Many toolkits for graph drawing are available, e.g. [10]–[12].

Another class of algorithms studies the visualization of dynamic graphs, e.g. see [13], [14] for recent surveys. Within the literature of dynamic graph visualization our method is most closely related to drawing superimposed node-link diagrams. Unlike most work in this direction, which is mainly theoretical, e.g. [15]–[19] and studies very small graphs whose edges do not intersect, we place special emphasis on robustness and efficiency of our techniques, without guaranteeing intersection-free results.

The papers more similar to our work, can be classified according to two design choices. The first choice is if the optimization computes graph layouts sequentially (e.g. [20], [21]) or jointly (e.g. [22], [23]). Sequential layouts are more suitable to graphs that are naturally ordered in a sequence. To compute a sequential layout, the node locations of the previous graph are considered as constraints for the node locations of the next graph in the sequence. For our problem, a joint layout computation is important. The second choice is if the optimization uses soft (e.g. [20], [24]) or hard (e.g. [22], [25]) constraints to layout nodes with the same labels. In our case hard constraints are not suitable, because our graphs typically have multiple nodes with the same label. That

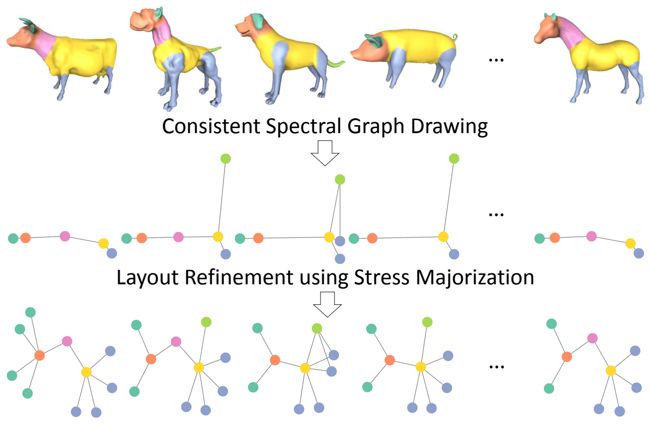

...
Consistent Spectral Graph Drawing
Layout Refinement using Stress Majorization

Fig. 2: **Framework Overview.** Top: A set of graphs is given as input. In this example the graphs are derived from segmented 3D shapes. Middle: the initialization given by the consistent spectral graph drawing algorithm. Bottom: the final consistent embedding with approximately preserved graph distances obtained with our consistent layout refinement algorithm using stress majorization.

would lead to degenerate layouts. In summary, we consider a joint layout problem using soft constraints, perhaps most similarly to the work of Erten et al. [4]. Below, we compare to a representative algorithm to show the advantages of our work. In contrast to all previous work, however, we can more systematically handle correspondences between groups of nodes and for the first time enable general linear combinations of nodes.

3 Joint Graph Layouts

The input to our method is a collection of graphs and partial correspondences between node sets of different graphs. To obtain consistent and structure preserving graph layouts we formulate an objective function that aims to produce layouts that both preserve the distances between nodes within individual graphs and, at the same time, to enforce consistency across the layouts of different graphs. There are two important challenges that our approach is aimed to overcome: first, in most practical settings we cannot rely on different graphs having the same number of nodes or even on the existence of precise node-to-node correspondences. Another important difficulty, which is already present in designing layouts of individual graphs and exacerbated in our setting is that objectives with both attractive and repulsive forces typically lead to non-linear non-convex energies, with potentially many local minima, making it difficult to find global (or even good local) optima.

To tackle these challenges we proceed in two stages (See Fig. 2). First, we formulate an easier problem, by generalizing the classical spectral graph drawing to the case of multiple graphs (See Section 3.2), which allows partial and soft correspondences. Our optimization algorithm considers the following two terms: the smoothness of each layout (forcing adjacent graph nodes to be laid out nearby) and the consistency of the layouts across different graphs. This generalization has the key property of admitting an efficiently computable globally optimal consistent embedding. However, spectral graph drawing can lead to cluttered layouts, since it lacks repulsive forces between nodes. Therefore, we provide an efficient layout refinement algorithm based on stress majorization (See Section 3.3). Our extension also allows partial and soft correspondences and improves upon the output of the spectral drawing method by aiming to preserve distances within individual graphs, while maintaining consistency across embeddings. Together, these two steps allow us to find a high quality solution extremely efficiently, making our method flexible and scalable to graphs with a significant number of nodes.

3.1 Problem Formulation

Input. We assume that we are given a collection of graphs $\{G_k\}_{k=1}^n$, where $G_k = (V_k, E_k)$ with $|V_k| = m_k$. V_k denotes the set of nodes and E_k denotes the set of edges. For each graph G_k we are given a possibly weighted adjacency matrix $A^k \in \mathbb{R}^{m_k \times m_k}$. An entry a_{ij}^k denotes the weight of an edge from node i to node j in graph k. If $a_{ij}^k = 0$ no edge exists between the corresponding nodes. Further, we assume to be given distances δ_{ij}^k between some pairs of nodes i and j in each graph k. The distances can be given as separate input, or derived from the adjacency matrix as graph distances, e.g. using Dijkstra's algorithm. For each pair of graphs (G_p, G_q), we are also given a set of c_{pq} correspondences, encoded by matrices $S_{pq} \in \mathbb{R}^{c_{pq} \times m_p}$ and $T_{pq} \in \mathbb{R}^{c_{pq} \times m_q}$, which encode relations between the nodes of different graphs.

Objective. Given this input, we would like to find the d-dimensional coordinates $X^{(k)} \in \mathbb{R}^{m_k \times d}$, where $1 \leq k \leq n$ and typically $d = 2$ or $d = 3$, such that:

1. For each graph, the distances δ_{ij}^k are approximately preserved.
2. For each pair of graphs (G_p, G_q), $S_{pq}X^{(p)} \approx T_{pq}X^{(q)}$.

We remark that our notion of relations between graphs encoded via matrices S_{pq}, T_{pq} is very general and allows to encode any linear dependencies between nodes or their

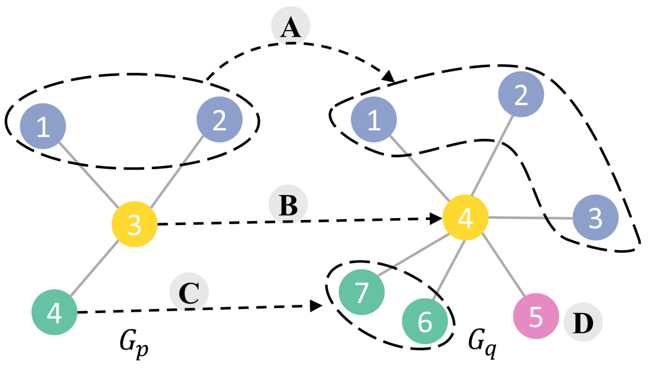


Fig. 3: **Correspondence Matrix.** Our goal is to find consistent layouts for the graph G_p and G_q (nodes are colored by labels and indexed by numbers), such that the nodes with the same label have consistent locations in these two graphs. Specifically, we may hope the (A) blue nodes $\{1, 2\}$ in G_p are mapped to the blue nodes $\{1, 2, 3\}$ in G_q, (B) node 3 in G_p is mapped to node 4 in G_q, (C) node 4 in G_p is mapped to nodes $\{6, 7\}$ in G_q, while (D) no constraint is added to node 5 in G_q.

groups. For example, it can easily accomodate precise node-to-node correspondences (in which case S_{pq} is a permutation matrix, whereas T_{pq} is identity), or partial probabilistic maps, which can be expressed using convex weights. Below we provide a few examples of pairs S_{pq}, T_{pq}, but we would also like to stress that throughout our algorithms *we do not place any restrictions on these matrices*, which contributes to the flexibility of our method.

We note also that although the two objectives lead to a non-linear non-convex energy, as we show below, it is possible to use a surrogate term for the second part of the energy, such that the global optimum can be obtained with spectral techniques, even though the formulation is not convex. This leads to a consistent spectral graph drawing approach (Section 3.2), which can in some simple cases be used by itself or as an initialization to the consistent stress majorization algorithm described in Section 3.3.

Example. For a better understanding of the modeling expressiveness, we describe a specific example using the graphs in Fig. 3. For this example, we denote the node positions as $\{x_i\}_{i=1}^4$ for graph G_p and as $\{y_j\}_{j=1}^7$ for graph G_q. In the first constraint the user expresses the idea that the set of blue nodes $\{1, 2\}$ in G_p are mapped to the set of nodes $\{1, 2, 3\}$ in G_q with a generalized linear equation. The second equation encodes that the node 3 in G_p is mapped to node 4 in G_q. In the third equation, the node 4 in G_p is mapped to a probability distribution over the nodes $\{6, 7\}$ in G_q. Node 5 in G_q does not appear in any equations. These location constraints can be formulated with the following linear equations:

$$\begin{aligned} a_1x_1 + a_2x_2 &\approx b_1y_1 + b_2y_2 + b_3y_3 \\ x_3 &\approx y_4 \\ x_4 &\approx py_6 + (1-p)y_7 \end{aligned} \tag{1}$$

The linear system of Eq. (1) can be equivalently formulated using matrices:

$$S_{pq}X \approx T_{pq}Y \tag{2}$$

where

$$S_{pq} = \begin{pmatrix} a_1 & a_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad T_{pq} = \begin{pmatrix} b_1 & b_2 & b_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & p & 1-p \end{pmatrix}$$

In the most general form $SX = TY$, constraints are encoded with a generalized linear map for arbitrary S and T. That means we can encode soft correspondences as a special case $SX = Y$ where S is a stochastic matrix (rows sum to one, all entries are non-negative).

3.2 Consistent Spectral Graph Drawing

Let L^k be the standard weighted Laplacian matrix associated with graph G_k. Using the notation above, the Laplacian can be computed as $L^k = \text{diag}(A^k\mathbf{1}) - A^k$, where $\mathbf{1}$ is the constant vector of all ones. The classical spectral graph drawing method [1], [26] consists of computing the d principal eigenvectors of L^k and using them as coordinate functions $X^{(k)}$ for drawing the graph G_k. Intuitively this method tries to find an embedding that is as smooth as possible so that adjacent vertices have similar coordinates (i.e., $\|L^k X^{(k)}\|_2$ is small) under the constraint that each coordinate function has a fixed unit norm, encoded by the constraint $(X^{(k)})^T X^{(k)} = Id$.

Our first goal is to extend this approach to the case of embedding multiple graphs with partial correspondences. For this we formulate the following problem:

$$\hat{X} = \underset{X^TX=I}{\arg\min}\; E_1(X) + E_2(X), \tag{3}$$
$$E_1(X) = \sum_{k=1}^{n} \|L^k X^{(k)}\|_F^2,$$
$$E_2(X) = \sum_{1\le p<q\le n} \|\mu_{pq}(S_{pq}X^{(p)} - T_{pq}X^{(q)})\|_F^2.$$

Here, X is constructed by stacking the matrices $X^{(k)}$ on top of each other.

Note that the first term in the energy tries to replicate the smoothness objective in classical spectral graph drawing, while the second term enforces the consistency of the embedding across multiple graphs, described earlier. For each correspondence we can optionally consider an importance parameter. For two graphs G_p and G_q the importance parameters are collected in a diagonal matrix $\mu_{pq} \in \mathbb{R}^{c_{pq}\times c_{pq}}$. For simplicity, we denote $B_{pq} = \frac{1}{2}\mu_{pq}S_{pq}$ and $D_{pq} = \frac{1}{2}\mu_{pq}T_{pq}$.

We also remark that the smoothness term can be alternatively formulated via the quadratic form $(X^{(k)})^T L^k X^{(k)}$. Our analysis below can easily be modified to incorporate such a term instead of the energy E_1, and we found the performance and layout quality to be similar with these two options.

Although the problem described in (3) seems significantly more involved than the one found in standard spectral graph drawing, we found an elegant reformulation of the optimization problem into a sparse eigen-decomposition problem for a larger matrix. In the paper, we only present the main result and defer the detailed derivation to the Appendix. In particular, consider the matrix $W \in \mathbb{R}^{m\times m}$,

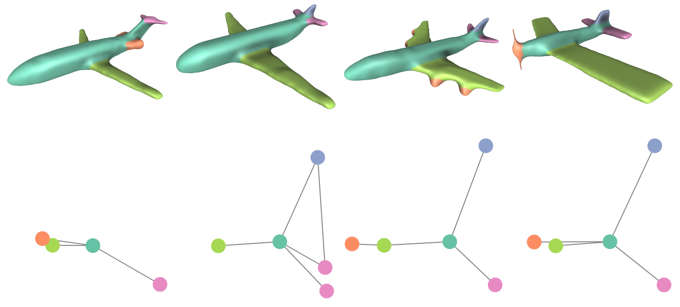


Fig. 4: Top: input shape collection, represented as graphs with nodes denoting the individual components of the shapes (colored) Bottom: joint embedding with consistent spectral drawing. Note that although the nodes with the same color are consistently positioned across graphs, each individual layout does not necessarily preserve graph structure well.

where $m = \sum_{k=1}^{n} m_k$. W is defined by blocks of $W_{pq} \in \mathbb{R}^{m_p\times m_q}$ as follows:

$$W_{pq} = \begin{cases} (L^p)^T L^p + \sum_{k\neq p}\left(B_{pk}^T B_{pk} + D_{kp}^T D_{kp}\right) & \text{if } p = q \\ -(B_{pq}^T D_{pq} + D_{qp}^T B_{qp}) & \text{otherwise} \end{cases}$$

The following proposition (proved in the Appendix) guarantees that the optimum embedding coordinates can be obtained by computing the principal eigenvectors of W.

Proposition 1. Assuming $\mu_{pq} = \mu_{qp}$, the globally optimal coordinates X for the energy in Eq (3) are given by the eigenvectors corresponding to the d smallest non-zero eigenvalues of W.

Figure 4 shows a result of consistent spectral drawing for a set of graphs for which exact node correspondences are not known due to label ambiguity. As can be seen in this figure, although the drawing obtained with this method is indeed consistent, it is nevertheless not very informative, since multiple nodes can potentially come into close proximity resulting in a very cluttered visualization. The primary reason for this phenomenon is that the energy described in (3) does not contain any terms aimed at preservation of graph distances in the embedding. In other words, it is possible that the result of spectral graph drawing is *too smooth* since there are no forces to push nodes apart. For this reason, in the following section we describe a refinement technique that tries to preserve the consistency of the drawing while enforcing the structural properties of individual graphs.

3.3 Layout Refinement using Stress Majorization

As mentioned above, one particular limitation of the energy function given in Eq. (3) is that it lacks terms that would maintain distances between nodes after the embedding. Therefore, we introduce another term to the energy with the goal of explicitly preserving the graph node distances. Thus, the optimal embedding $\hat{X}$ should minimize the following objective function:

$$F(X) = E_1(X) + E_2(X) + E_3(X), \tag{4}$$

$$E_3(X) = \sum_{k=1}^{n} \sum_{1 \le i < j \le m_k} \lambda_{ij}^{(k)} \left(\|X_i^{(k)} - X_j^{(k)}\| - \delta_{ij}^{(k)} \right)^2,$$

where $\delta_{ij}^{(k)}$ are graph distances between the i-th and j-th node in the graph G_k and $\lambda_{ij}^{(k)}$ are scalar weights which give us control over the importance of the preservation of distances between each pair of nodes in each graph.

To find the optimal consistent embeddings $\hat{X}$ of Eq. (4), one possibility would be to use standard non-linear optimization approaches, such as BFGS or other quasi-Newton techniques. Note, however, that the energy might not be differentiable if some nodes have the same coordinates. Moreover, the computation of the gradient can be expensive and standard techniques can easily get trapped in shallow local minima.

Therefore, following [27] we use a convex majorization technique which is both efficient and produces high quality results. This convex majorization technique is commonly used for optimizing the metric Multidimensional Scaling (MDS) energy (in a similar form of E_3 in Eq. (4)), which is proposed by de Leeuw [7] and applied to graph drawing by Gansner [3]. We extend the existing problem formulation by additional terms (E_1 and E_2 from Eq. (3)) related to smoothness of the layout of each graph and consistency of the layouts of different graphs. We built on de Leeuw's work to derive a stress majorization algorithm based on the Cauchy-Schwartz inequality for our extended problem formulation (See Appendix B for the derivation).

Specifically, we first introduce a majorizing function $g(X, Z)$ for our objective function $F(X)$ given in Eq. (4), such that $g(X, Z)$ is quadratic in X given any fixed Z, $F(X) \le g(X, Z)$, and $F(Z) = g(Z, Z)$. The supporting point Z_k will be updated iteratively to provide a descent direction for $F(X)$, and the algorithm is guaranteed to converge to a local minimum.

Specifically, following [27] we propose the following optimization procedure:

1 Set $Z = Z_0$, where Z_0 is initialized by the Spectral Layout.
2 Update X^u by solving the linear system given in Algorithm 1
3 If $F(Z) - F(X^u) < \epsilon$, then stop.
4 Set $Z = X^u$ and go to 2.

Note that not all objective functions admit a majorizing function that is easy to optimize. However, as shown in the following proposition (proved in the appendix) such a function can be obtained for the energy function $F(X)$ given in Eq. (4).

Proposition 2. There exists a majorizing function $g(X, Z)$ for the energy given in Eq. (4) such that the optimization of $g(X, Z)$ with a fixed Z can be done using a solution of a single linear system of equations.

This proposition ensures that even though Eq. (4) leads to a non-convex non-linear optimization problem, it can nevertheless be optimized efficiently by solving a sequence of convex problems. Despite this, both in theory and in practice, having a good initialization is fundamental to get a high quality result efficiently. Therefore, we use the solution found with the approach described in Section 3.2 and refine it with the stress majorization method described above. This two-stage approach allows us to avoid undesirable local minima of Eq. (4) and at the same time maintain scalability of the method, making it capable of processing collections of graphs potentially with thousands of nodes. Fig. 5 illustrates the iterative process of the stress majorization algorithm initialized by the spectral drawing algorithm (the second column).

In summary, the algorithms described in this section can compute joint graph layouts for a set of graphs with given correspondences. In the next section, we introduce a topological shape space exploration framework that uses these joint graph layouts for visualization.

Algorithm 1: Update rule for our layout refinement algorithm with stress majorization.

```
1) For every graph $k$ construct the matrix $L_\lambda^k$ s.t.
     $(L_\lambda^k)_{ij} = -\lambda_{ij}^{(k)}$ if $i \neq j$ and $\sum_{j \neq i} \lambda_{ij}^{(k)}$ otherwise.
2) For every graph $k$ and every pair of graphs $p, q$
     construct:
     $M_k = (L^k)^T L^k + \frac{1}{2} \sum_{p \neq k} (D_{pk}^T D_{pk} + B_{kp}^T B_{kp}) + L_\lambda^k$
     $N_{pq} = -B_{pq}^T D_{pq} - D_{qp}^T B_{qp}$
     $Q_{pq} = \frac{1}{2} (B_{pq}^T B_{pq} + D_{qp}^T D_{qp})$
3) Contruct the matrix $V$ given blockwise as:
     $V_{pq} = \begin{cases} M_p + M_p^T + \sum_{k \neq p} (Q_{pk} + Q_{pk}^T) & \text{if } p = q \\ N_{pq} + N_{qp}^T & \text{otherwise} \end{cases}$
4) Construct the block-diagonal matrix $U$ with the $k$-th
     block given by the matrix $W^{Z^{(k)}}$ where
     $W_{ij}^{Z^{(k)}} = -\frac{2\lambda_{ij}^{(k)} \delta_{ij}^{(k)}}{\max(\epsilon', \|Z_i^{(k)} - Z_j^{(k)}\|)}$ if $i \neq j$
     and $\sum_j -W_{ij}^{Z^{(k)}}$ otherwise.
5) Solve the linear system of equations $VX^u = UZ$.
```

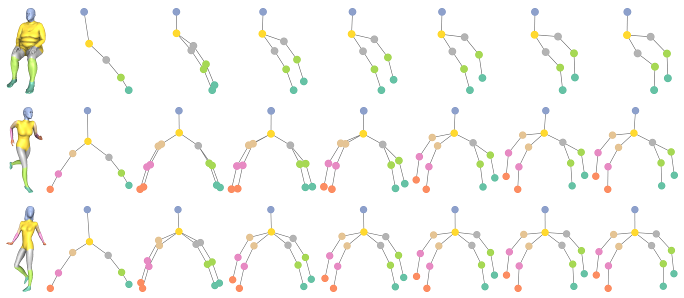


Fig. 5: **Stress Majorization Refinement Iterations.** The first column shows the human shapes from the Princeton Shape Benchmark; the second column shows the consistent spectral drawing initialization; the following columns are the result from the 5th, 8th, 10th, 15th, 30th and 50th iterations respectively ($c_1 = 2, c_2 = c_3 = 100, c_4 = 0.6$).

4 Topological Shape Space Exploration

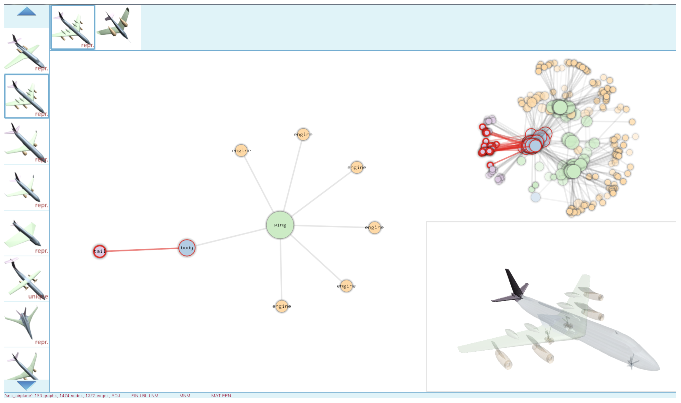


Fig. 6: **User Interface.** In the left sidebar we show a set of 3D models represented by pre-computed 3D renderings. In the top bar, we show pre-computed 3D renderings of all models that have the same topology as the currently selected one. The main view shows the graph of the selected model while the lower right shows an interactive 3D rendering. The upper right shows an overlay of topologically similar models. The selected node (pink in the main view) is propagated both to the overlay view as well as the interactive 3D view (the tail is highlighted).

4.1 User Interface

We developed a user interface that enables exploration of collections of 3D models based on the consistent graph embedding algorithms. We have a flexible user interface that provides multiple views of the data. These views are shown in subwindows that can be rearranged. See Fig. 6 for an example configuration. We provide the following views: 1) A mesh view, showing a single rendered 3D mesh. 2) A scrollable list showing renderings of multiple 3D models. 3) A view for a single or multiple graphs in the same coordinate system. 4) A grid view showing a set of graphs side-by-side.

We provide standard user interface elements to control the views such as scrolling, zooming, and panning. We also provide standard selection tools, e.g., for selecting models, nodes and edges.

We also provide multiple types of parameters for the visualization, such as setting node sizes, labeling, and coloring nodes according to a computed distortion function.

Our interface also provides meaningful links between the different views. For example, the grid views are linked so that all the graphs can be zoomed and translated simultaneously. Another example is that selecting a graph will automatically update the mesh exploration view to show the corresponding model. We can also select a component in the 3D mesh view which results in highlighting the corresponding node of the graph. An important feature is the propagation of selections.

We also include basic tools to select a subset of graphs, e.g. based on the number of nodes or the number of nodes of a particular type. One important graph selection tool is to select graphs based on the similarity to a selected source graph. The resulting list is typically processed in a sorted fashion.

For navigation, we offer different possibilities. 1) Given a sequence of graphs, the user can cycle through graphs in the sequence. The transition can either be hard switching or a smooth transformation. 2) A user can navigate from a currently selected graph to similar graphs in the collection.

Please see the accompanying video for a demonstration of the user interface.

4.2 Preprocessing

We take a collection of segmented and labeled 3D meshes as input and convert each mesh to a graph representation. We show two types of meshes in this paper: manifold triangle meshes from the Princeton Shape Benchmark [28] and CAD models from the Stanford ShapeNetCore dataset [29]. For each mesh, we first create connected components by merging connected triangles with the same label. Then we create a labeled node for each component and create an edge with weight one between all components that are directly connected (or intersecting). For the Princeton Shape Benchmark we use the provided segmentation and labels. For the ShapeNetCore dataset the per-face labeling comes from [30]. A common problem with CAD meshes are topological inconsistencies, such as incomplete connectivity, self-intersections, duplicate faces, etc. The Stanford ShapeNetCore dataset is not free from such defects, resulting in multiple sets of small topological components that each form one larger semantic unit. We therefore perform a proximity-based topological cleanup. We detect pairs of components that intersect or are closer to each other than a small threshold. We merge a pair of such components if they share the same label. Otherwise, we keep them separated but add an edge between the corresponding nodes in the graph.

Moreover, we establish correspondences between nodes with the same labels across different graphs. The particular challenge of these datasets is that each graph can contain multiple components with the same label and that there is no clear one to one correspondence between nodes. We use probabilistic maps (soft maps) with uniform distribution to construct the generalized linear correspondences S and T.

5 Results

All the experiments are performed on a workstation with a 3.10GHz processor and 64 GB memory. The graph processing algorithms are implemented in MATLAB and the user interface is implemented in C++ using OpenGL.

To evaluate our framework, we tested our method on five datasets: the Princeton Shape Benchmark [28], the Stanford ShapeNetCore dataset [29], the Stanford Scene database [31], floor plans and food networks [32]. Table 1 shows the graph complexity of each dataset and the corresponding runtime performance.

Specifically, we tested six model categories (Airplane, Ant, Armadillo, FourLeg, Human and Teddy) from the Princeton Shape Benchmark [28]. Each category has twenty different shapes. We also tested our methods on four shape categories (Rocket, Motorcycle, Car, and Airplane) from the Stanford ShapeNetCore dataset [29]. Specifically, there are 66 rockets, 198 motorcycles, 474 cars, and 2035 airplanes.

Dataset		Total No. of Graphs	No. of Unique Labels	Total No. of Nodes	Total No. of Edges	Running time (s)			Initial Energy ($\times 10^4$)	Final Energy ($\times 10^4$)	No. of Iterations
						Pre-process	Spectral Drawing	Stress Majorization			
PSB	Airplane	20	5	125	106	0.0624	0.1004	0.2499	3.2077	0.1286	77
	Ant	20	5	220	200	0.0698	0.1048	0.2625	9.6259	0.5671	81
	Armadillo	20	11	337	365	0.1063	0.1216	0.5591	27.099	1.3486	111
	FourLeg	20	6	190	172	0.0949	0.1107	0.3767	8.2606	0.4849	125
	Human	20	8	271	260	0.1000	0.1156	0.4932	16.481	0.7491	128
	Teddy	20	5	160	142	0.0951	0.1079	0.1662	5.6188	0.2085	50
SNC	Rocket	66	3	286	238	0.2614	0.2151	1.1059	27.937	1.6932	141
	Motorcycle	198	6	1340	1552	1.3556	1.3092	10.031	195.43	15.235	123
	Car	474	4	2017	1576	6.4641	6.1177	53.168	176.94	29.773	222
	Airplane	2035	4	14978	17201	110.75	144.08	1397.0	5116.2	1198.2	50
Scenes		6	31	74	94	0.0615	0.1011	0.1185	4.2547	0.4043	99
Floor plans		4	22	84	88	0.0579	0.0918	0.1244	8.3810	0.7013	116
Food network		4	75	300	9494	0.0982	0.1220	3.9745	107.85	71.793	781

TABLE 1: **Final Result.** A summary for each dataset, including the Princeton Shape Benchmark (PSB), the Stanford Scene database, the Stanford ShapeNetCore dataset (SNC), floor plans, and food networks; the performance measurements for the experiments on each dataset, including the running time for each step, the initial and final energy, and the total number of iterations until convergence.

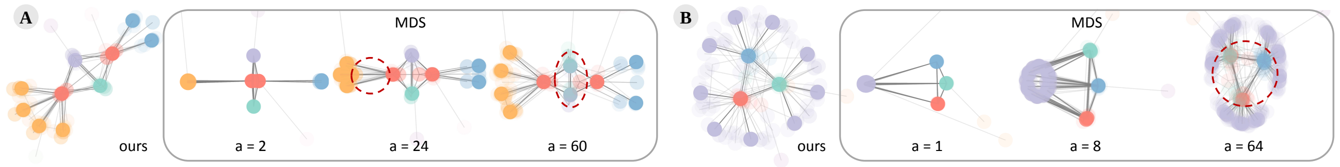


Fig. 7: **Illustration of the failure of the MDS-based method**. Here, the parameter a is the ratio between the weight of graph-distance preservation and the weight of consistency preservation. When a is small (e.g. $a = 2$ for collection A and $a = 1$ for collection B), the graphs are consistent, but the information about the graph structure is almost completely lost. When a is large (e.g., $a = 60$ for collection A and $a = 64$ for collection B), the graph distance for each graph is well-preserved. However, the consistency among the graphs is broken. For collection A, the light-green and purple nodes (in the red dashed circle) are mixed together and similarly for the nodes with color red, green and blue in collection B. Even after tuning the parameter a per visualization to give a balance between the distance and consistency preservation (e.g. $a = 24$ for collection A and $a = 8$ for collection B), it is not possible to create a good result.

In terms of running time (see Table 1), the first step, which consists of the consistent spectral graph drawing, is obtained by solving a simple sparse eigen-decomposition problem. Therefore, the time spent on the first step is correlated with the total number of graphs and nodes but remains efficient for graphs with hundreds or even thousands of nodes. The time spent on the second step using stress majorization depends on the graph complexity and the tradeoff among different terms in the energy function. We show visualizations of the datasets throughout the paper and also in the accompanying video and additional materials. More details about the Stanford Scence database, floor plans, and food networks is provided in the additional materials.

5.1 Evaluation of the Problem Formulation

To evaluate our problem formulation, we compare to two alternative graph embedding methods: (1) embedding each graph individually using Kamada-Kawai algorithm [2]; (2) jointly embedding the graphs using MDS [3] with virtual edges added between nodes of different graphs that have the same label [4]. We choose this approach, because it uses the same distance preservation term as our method. Therefore, the comparison can better highlight the difference between our consistency term and virtual edges.

We compiled a list of analysis tasks for exploring a collection of graphs and designed a list of representative questions to evaluate the different algorithms:

Q1 Are the graphs in the collection the same or not?

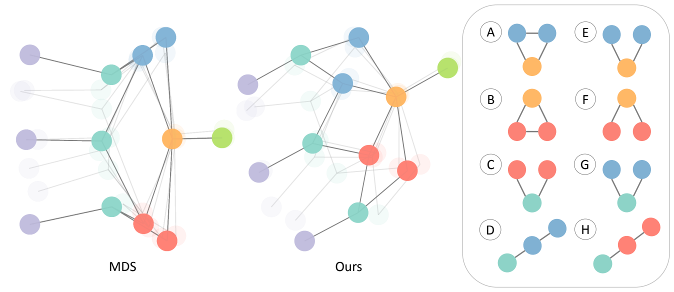


Fig. 8: **Sample Task Q7.** The user is given a collection of graphs in an overlaid fashion where the graphs are drawn jointly using either the MDS-based method or our method (randomly chosen). Then the user is asked to select from (A-H) the subgraphs that appear in the dominant structure of the given collection. The purpose is to measure how well the user understands the graph collection with the given embedding, which can be further used as a measure of the usefulness of the embedding. In this example, the correct answers are (C,E,F,G). Fig. 9 shows that using our method, 80% of the users can give the correct answer while only 43% of the users give the correct answer using the MDS-based method.

Q2 Which graph is different from the rest?
Q3 Which collection has a larger variability?
Q4 What is the dominant structure of this collection?
Q5 Nodes with what label have a larger variability?
Q6 How many orange nodes do most of the graphs have?

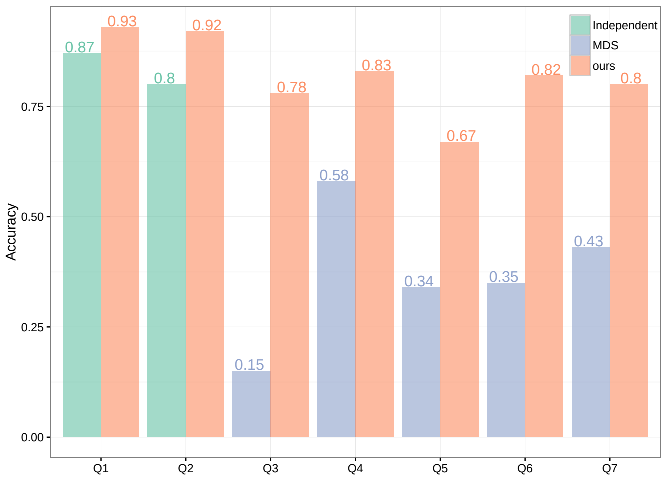


Fig. 9: **Average Accuracy.** The average accuracy over 104 users and two independent datasets for each task. *Independent*: the graphs in the collection are drawn independently using the Kamada-Kawai method. *MDS*: the graphs are drawn jointly using the MDS method. *ours*: the graphs are drawn jointly using our method.

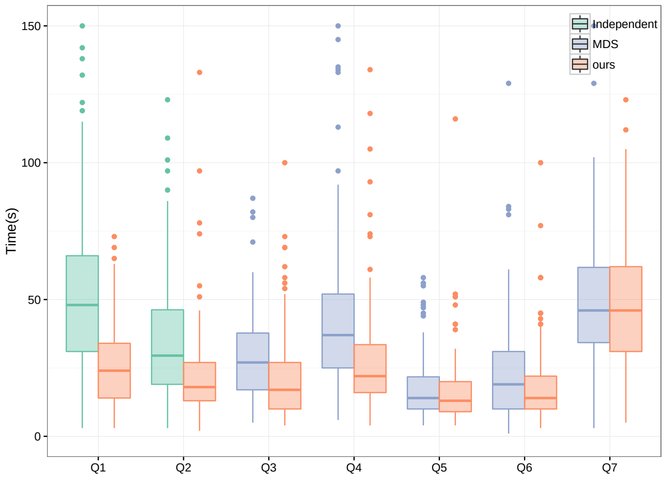


Fig. 10: **Time**. The summary statistics ($\frac{1}{4}$, $\frac{1}{2}$, $\frac{3}{4}$ quartiles and outliers) of time the user spent over 104 users and two independent datasets for each task. *Independent*: the graphs in the collection are drawn independently using the Kamada-Kawai method. *MDS*: the graphs are drawn jointly using the MDS method. *ours*: the graphs are drawn jointly using our method.

Q7 Which subgraphs appear in the dominant structure of the given collection?

In Fig. 8 we show an example of the kind of a question from the last category (Q7).

To remove the bias, we used different graph collections for different tasks and the order of the selections for each task is randomized for each test. We tested each task on two independent datasets and collected responses from 104 users using an internet-based survey tool. Fig. 9 shows the average accuracy and Fig. 10 shows the summary statistics of the time that the users spent for different tasks with different methods. Specifically, with pre-aligned graphs (using our method), it takes less time to get a higher accuracy compared to the graphs without alignment (using the Kamada-Kawai for each graph independently). We can also observe that our method achieves much higher accuracy within similar or even less time compared with the MDS method where the graph distances fail to be preserved properly. Therefore, the results provide a strong indication for the importance of consistent alignment and graph distance preservation for understanding a collection of graphs.

Dataset	Final Energy ($\times 10^4$)		Runtime (s)	
	BFGS	Stress Major.	BFGS	Stress Major.
Airplane	0.2387	**0.1286**	62.378	**0.6871**
Ant	1.6183	**0.5671**	90.615	**0.8114**
Armadillo	**1.3480**	1.3486	404.94	**1.3807**
FourLeg	1.5759	**0.4849**	105.85	**1.1918**
Human	0.7683	**0.7491**	283.95	**1.2646**
Teddy	0.2085	0.2085	208.40	**0.4675**
Scenes	**0.3882**	0.4043	27.279	**0.2330**
Floor plans	0.7137	**0.7013**	19.510	**0.2439**
Food network	73.7916	**71.7925**	207.89	**5.8086**
Rocket	6.1400	**1.6934**	36291	**4.8124**

TABLE 2: The final energy and the runtime of using the BFGS and our stress majorization: our approach takes less time to reach a lower final energy.

Fig. 7 gives two specific examples for the comparison between the MDS-based method and ours. Both collection A and B have around 50 graphs with similar structures and the graphs are drawn jointly in an overlaid fashion. For the MDS-based method, the parameter a is the ratio between the weight of graph-distance preservation and the weight of consistency preservation. We can see that it is hard for MDS-based method to balance the graph distance preservation and the consistency. As a result, all possible parameter settings for the parameter a lead to artifacts in the MDS-based layouts. To avoid problems due to parameter settings as much as possible, we tuned the parameters per visualization for the MDS-based method, but used the default parameters for our method. Even though this gives an advantage to the MDS-based method in the user study, the results of our method are still better.

We also show a comparison with the MDS-based method on large datasets: the Stanford SNC dataset. Fig.11 shows (A) rockets, (B) motorcycles, and (C) airplanes from the Stanford SNC dataset using the metric MDS (left) and our joint layout framework (right). For the metric MDS method, a distance matrix is constructed for the whole graph collection with added virtual edges such that (1) the distance between the two nodes from the same graph is the consistent with their original graph distance; (2) the distance to the nodes from different graphs with the same label is set to zero; (3) the distance-preserving weight for the nodes from different graphs with different labels is set to zero such that these nodes pairs will be ignored in the MDS stress energy. The results show that the metric MDS layout does not depict important information that is clearly visible in our joint layouts. See the figure caption for the description of selected examples.

5.2 Evaluation of the Optimization Algorithm

We conducted a comparison between our proposed stress majorization algorithm with BFGS to optimize the energy

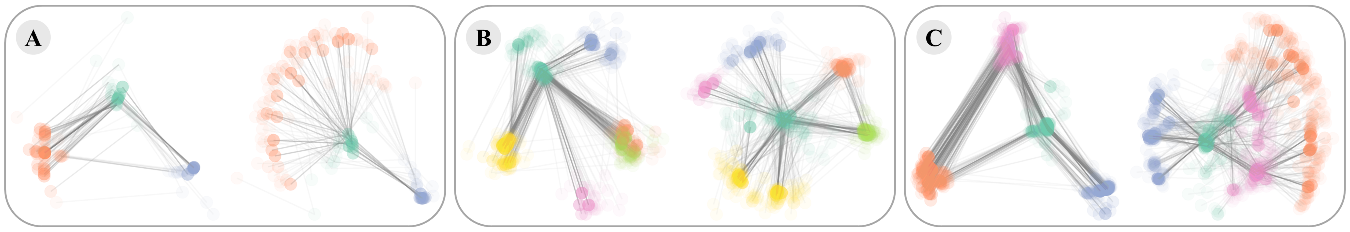


Fig. 11: **Comparison to MDS.** The metric MDS layouts (left) and our joint layouts (right) for (A) rockets, (B) motorcycles, and (C) airplanes from the SNC dataset (nodes colored by labels). The MDS layouts cannot preserve important aspects of the graph structure that are clearly visible with joint layouts. In (A) joint layouts show that rockets have multiple fins (colored orange) and that there is a large variety in the number of fins across the dataset. In (B) joint layouts show that motorcycles generally have two wheels (colored yellow) and that the gas tank (orange) is often connected to the seat (light green). In (C) joint layouts show the airplanes often have two wings (purple) and that there is variability in the number of engines (orange) across the dataset.

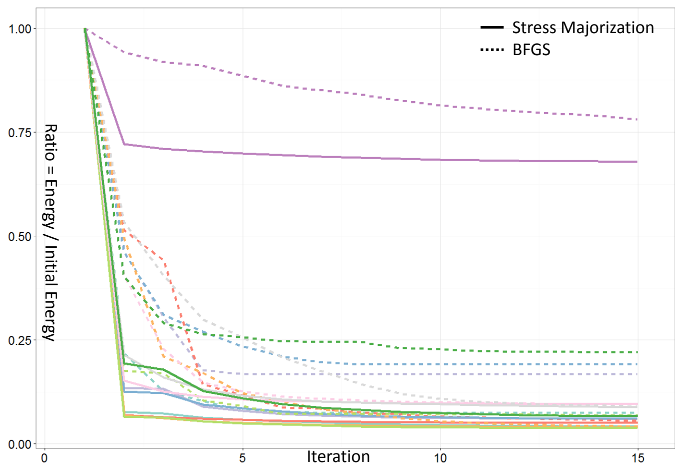


Fig. 12: Energy ratio per iteration: the energy (normalized by the initial energy) per iteration from ten datasets using two different methods, the BFGS (dashed lines) and the stress majorization (solid lines).

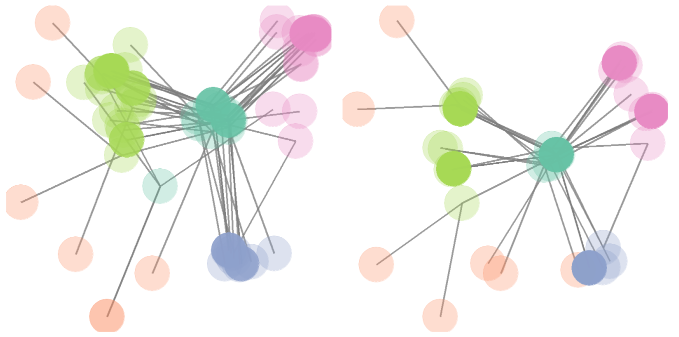


Fig. 13: Overlaid 20 Airplanes from the Princeton Shape Benchmark: the BFGS method (left) and the stress majorization method (right).

function Eq. (4). The result shows that the stress majorization has a better performance over the BFGS method.

Fig. 12 shows the energy ratio per iteration of the BFGS and the stress majorization method, where the energy ratio is a percentage with respect to the initial energy indicating the convergence speed. Ten datasets (six datasets from the PSB, one from the Stanford SNC, the scene, the floor plan, and the food network) are tested. The corresponding final energy and runtime of the BFGS and the stress majorization method are recorded in Table 2.

Table 2 shows that the stress majorization takes much less time to reach a smaller final energy compared to BFGS. Fig. 12 shows that the stress majorization (solid lines) converges faster than the BFGS method (dashed lines). Fig. 13 gives an example of the joint layouts using BFGS (left) and the stress majorization (right) respectively on the Airplane dataset from the Princeton Shape Benchmark. The BFGS method leads to a layout where the components with different labels are separated from each other, but the stress majorization gives a better overview of the airplane structure where the two independent stabilizers (colored purple) and the wings (colored light green) get separated as well.

5.3 Comparison to Random Initialization

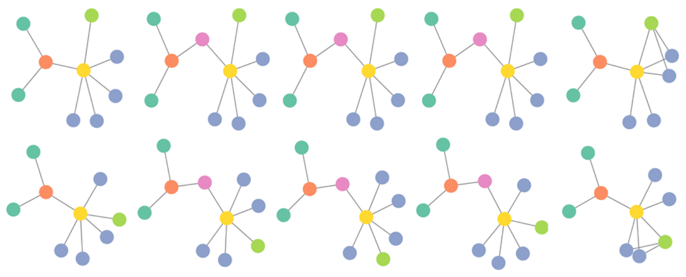


Fig. 14: **Usefulness of the Spectral Drawing Initialization.** The first row is given by the spectral drawing initialization with a lower final energy compared with the second row which is initialized by a random layout ($c_1 = 2, c_2 = c_3 = 100, c_4 = 0.8$). The spectral drawing initialization leads to a better layout where the light green nodes corresponding to the tail are more consistent with each other.

To evaluate the usefulness of the spectral drawing initialization, we compared the layout refinement using stress majorization using random initialization and spectral drawing initialization. The corresponding statistics are shown in Table 3. For each of the collections (Airplane, Ant, Armadillo, FourLeg, Human and Teddy), the layout refinement step is initialized randomly 100 times. We report the corresponding number of iterations until convergence, running time, as well as the initial and final total energy. In most cases, the spectral drawing initialization outperforms the random

	Measure	Random Initialization				Spectral Drawing Ini.
		Min.	Med.	Max.	Ave.	
Airplane	No. Iter.	87	153	519	169	77
	Time (s)	0.155	0.270	0.812	0.301	0.154
	Ini.(10^4)	3.674	3.735	3.773	3.735	3.208
	Final (10^4)	0.129	0.137	0.320	0.183	0.129
Ant	No. Iter.	73	107	400	113	81
	Time (s)	0.225	0.326	1.113	0.339	0.251
	Ini.(10^4)	10.21	10.29	10.34	10.29	9.626
	Final (10^4)	0.567	0.567	0.726	0.569	0.567
Armadillo	No. Iter.	100	186	656	206	111
	Time (s)	0.557	0.993	3.460	1.114	0.616
	Ini.(10^4)	27.99	28.07	28.13	28.07	27.10
	Final (10^4)	1.358	1.392	1.544	1.408	1.349
FourLeg	No. Iter.	85	147	405	160	125
	Time (s)	0.244	0.405	1.046	0.436	0.347
	Ini.(10^4)	8.705	8.763	8.814	8.761	8.261
	Final(10^4)	0.485	0.491	0.672	0.549	0.485
Human	No. Iter.	75	189	702	209	128
	Time (s)	0.303	0.717	2.576	0.794	0.501
	Ini.(10^4)	17.22	17.29	17.34	17.29	16.48
	Final (10^4)	0.751	0.777	1.007	0.809	0.749
Teddy	No. Iter.	64	106	368	127	50
	Time (s)	0.142	0.219	0.708	0.259	0.119
	Ini.(10^4)	6.193	6.236	6.287	6.239	5.619
	Final(10^4)	0.209	0.463	0.515	0.370	0.209

TABLE 3: **Usefulness of Spectral Drawing Initialization.** The statistics of running the stress majorization algorithm initialized randomly 100 times and the performance with the spectral drawing initialization as a comparison: the spectral initialization leads to a better result with a smaller number of iterations, a smaller amount of runtime and a lower final energy compared with the random initialization.

initialization with respect to the final energy. Even a small difference in the energy can indicate an undesirable inconsistency in the layout. Fig. 14 shows such an example of the final embedding computed based on the spectral drawing initialization and a random initialization. The first row is given by the spectral drawing initialization which has a lower final energy since the nodes corresponding to the tail (shown in light green) are highly consistent with each other, while in the final embedding given by the random initialization they are not. Therefore, the spectral drawing step helps to separate the nodes with different labels and enforce the consistency.

5.4 Choice of Parameters

In our experiments we consider the graph distances $\delta_{ij}^{(k)}$ to be simply the lengths of the shortest paths within graph G_k. In the case of weighted graphs (such as the ones in the food networks), we take *edge length* $=$ $1/$*edge weight*. We use probabilistic maps (soft maps) with uniform distribution to construct the generalized linear correspondences S and T. The matrix μ_{pq} (see Eq. (3)) is constructed as a global constant c_1 times the identity matrix in our results. Alternatively, each correspondence could be weighted by the number of nodes involved. We also have a scalar weight $\lambda_{ij}^{(k)}$ (see Eq. (4)) to control the importance of distance preservation between the i-th and j-th node in the graph G_k. In the classical graph drawing algorithm [2], this parameter is given by $\lambda_{ij}^{(k)} \propto 1/\delta_{ij}^{(k)2}$. In our experiments, we set $\lambda_{ij}^{(k)} = c_2/{\delta_{ij}^{(k)}}^2 + c_3 a_{ij}^k$ where A^k is the adjacency matrix for graph G_k. Intuitively, we try to preserve the graph distances for those pairs of nodes that are close to each other or directly connected to each other. Moreover, since we have the labels for the nodes, we can encode this information into the graph distances. We introduce another parameter $c_4 \in (0, 1]$ to update the graph distance ${\delta_{ij}^{(k)}}' = c_4 \delta_{ij}^{(k)}$ if i-th node and j-th node have the same label in the graph G_k. In this case, we are expecting the nodes with the same label to be closer to each other. Further, we use $\epsilon = 10^{-3}$ as the convergence criterion for the stress majorization algorithm.

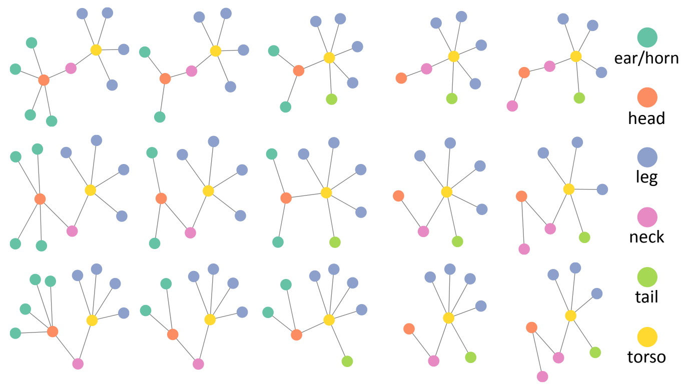


Fig. 15: **The influence of different parameters.** The first row, $c_1 = 1, c_4 = 1$; the second row, $c_1 = 5, c_4 = 1$; the last row, $c_1 = 5, c_4 = 0.8$. The first two rows show that a larger c_1 will give a more consistent layout; the last two rows show that the nodes with the same label will be closer to each other with a smaller c_4.

Fig. 15 shows results for different parameters. We set $c_2 = c_3 = 100$ for these three tests. When comparing the first two rows, the top one has a weaker consistency strength μ. We can see that in the second row the nodes that represent the head (colored in orange) and the torso (colored in yellow) are much more consistent than the first row. In this case, we could keep a balance between the consistency term and distance preservation term by controlling the corresponding parameters. Comparing the last two rows, we see that the nodes with the same label stay close to each other while the node representing the tail (with color green) gets separated from the cluster of legs (with color steel blue) if we set c_4 to a smaller value.

5.5 Limitations

One limitation of our method is that the spectral initialization requires a sparse Eigen-decomposition and the stress majorization process requires linear system solving for each iteration, which might limit scalability, although our complexity remains manageable even for collections containing several thousand graphs with tens of thousands of nodes total.

5.6 Joint graph matching by consistent layout

We also ran a series of experiments in order to validate the output of our algorithm, by considering its performance with incomplete correspondence information. These experiments show that even with a fraction of the input, our approach produces results that are comparable to having

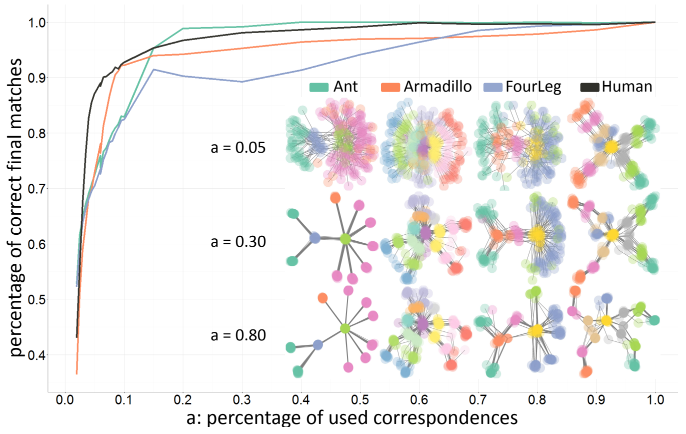


Fig. 16: **Partial Correspondence.** The average on the percentage of correct final matches over 50 random tests given each a, the percentage of used correspondences. The bottom-right figure shows the overlaying final embedding of the tested models for three specific choices of a.

complete correspondences across graphs. This suggests that both the energy and the optimization technique are consistent with the underlying (possibly unknown) joint graph information.

Specifically, we tested the possibilities to discover node correspondences and labeling across a collection of graphs given incomplete and approximate matches, by using our framework. For this we randomly removed a fraction a of correspondence constraints from the given consistency mapping matrices S and T, and used our algorithm to find the final layout. We then constructed the full node-to-node correspondences by taking the label from the nearest neighbor for each pair of graphs after the final layout. Fig. 16 shows the results of four models (Ant, Armadillo, Fourleg and Human) from the Princeton Shape Benchmark. The bottom-right figure shows the joint embedding of all the graphs for three different choices of a. We then measured the number of correct final matches as a measure compared to the number of input correspondences. We can see that in most models even with less than 10% of the matches, the number of correct correspondences is between 80 and 90%, while for the model Ant, the overall correspondences could be fully recovered on average given only 20% node matches. Generally, the more information of node correspondences is provided, the more consistent the final layout will be. When comparing different models, we can find that the more diverse the graphs are, the more correspondences we need to use to make them consistent.

6 Conclusion and Future Work

In this paper we introduced a novel and efficient approach for computing joint graph layouts and show applications to visualizing collections of segmented and labeled 3D meshes. Our main contributions are a spectral graph layout algorithm and a layout refinement algorithm using stress majorization. We demonstrated the usefulness of our framework on a variety of datasets including the Princeton Shape Benchmark, the Stanford ShapeNetCore dataset, the Stanford Scene database, floor plans, and food networks.

We believe that there are several interesting questions for future work. First of all, the consistency mapping in our setting relies on prior knowledge, i.e. labeled data. In the future, we would like to explore algorithms to automatically find correspondence (encoded by matrices S and T). Further, we would like to investigate hierarchical extensions of our work to graphs containing millions of nodes. We are also interested in studying the applicability of our techniques for dynamic and time-varying graph collections. Finally, we are planning to investigate how the approximate symmetries of the input data can both guide the joint embedding and can conversely be extracted from it.

Acknowledgements

M. Ovsjanikov's work is funded through the following grants: Marie-Curie CIG-334283, a CNRS chaire d'excellence, chaire Jean Marjoulet from École Polytechnique, FUI project TANDEM 2, and a Google Focused Research Award. J. Ren, J. Schneider and P. Wonka are funded through the Visual Computing Center (VCC) at King Abdullah University of Science and Technology (KAUST).

References

[1] Y. Koren, "On spectral graph drawing," in *Computing and Combinatorics*. Springer, 2003, pp. 496–508.
[2] T. Kamada and S. Kawai, "An algorithm for drawing general undirected graphs," *Information Processing Letters*, vol. 31, no. 1, pp. 7 – 15, 1989.
[3] E. R. Gansner, Y. Koren, and S. North, "Graph drawing by stress majorization," in *International Symposium on Graph Drawing*. Springer, 2004, pp. 239–250.
[4] C. Erten, S. G. Kobourov, V. Le, and A. Navabi, "Simultaneous graph drawing: Layout algorithms and visualization schemes." *J. Graph Algorithms Appl.*, vol. 9, no. 1, pp. 165–182, 2005.
[5] P. Eades, "A heuristics for graph drawing," *Congressus numerantium*, vol. 42, pp. 146–160, 1984.
[6] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Softw., Pract. Exper.*, vol. 21, no. 11, pp. 1129–1164, 1991.
[7] J. De Leeuw, "Applications of convex analysis to multidimensional scaling," *Department of Statistics, UCLA*, 2005.
[8] L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality reduction: a comparative review," *J Mach Learn Res*, vol. 10, pp. 66–71, 2009.
[9] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J.-D. Fekete, and D. W. Fellner, "Visual analysis of large graphs: State-of-the-art and future challenges," *Computer Graphics Forum*, 2011.
[10] J. Ellson, E. R. Gansner, E. Koutsofios, S. C. North, and G. Woodhull, "Graphviz and dynagraph – static and dynamic graph drawing tools," *Graph drawing software*, pp. 127–148, 2004.
[11] E. R. Gansner, "Drawing graphs with graphviz," *Technical Report, Technical Report*, 2009.
[12] M. Jünger and P. Mutzel, *Graph drawing software*. Springer Science & Business Media, 2012.
[13] F. Beck, M. Burch, S. Diehl, and D. Weiskopf, "A taxonomy and survey of dynamic graph visualization," in *Computer Graphics Forum*. Wiley Online Library, 2016.
[14] P. J. Sazama, "An overview of visualizing dynamic graphs," 2015.
[15] P. Brass, E. Cenek, C. A. Duncan, A. Efrat, C. Erten, D. P. Ismailescu, S. G. Kobourov, A. Lubiw, and J. S. Mitchell, "On simultaneous planar graph embeddings," *Computational Geometry*, vol. 36, no. 2, 2007.
[16] S. Cabello, M. J. van Kreveld, G. Liotta, H. Meijer, B. Speckmann, and K. Verbeek, "Geometric simultaneous embeddings of a graph and a matching." *J. Graph Algorithms Appl.*, vol. 15, no. 1, pp. 79–96, 2011.
[17] A. Estrella-Balderrama, J. J. Fowler, and S. G. Kobourov, "Graphset, a tool for simultaneous graph drawing," *Softw. Pract. Exper.*, vol. 40, no. 10, pp. 849–863, 2010.

[18] J. Cappos, A. Estrella-Balderrama, J. J. Fowler, and S. G. Kobourov, "Simultaneous graph embedding with bends and circular arcs," *Computational Geometry*, vol. 42, no. 2, pp. 173 – 182, 2009.

[19] T. Bläsius, S. G. Kobourov, and I. Rutter, "Simultaneous embedding of planar graphs," in *Handbook of Graph Drawing and Visualization*. CRC press, 2013, ch. 11, pp. 349–374.

[20] M. Gaertler and D. Wagner, *A Hybrid Model for Drawing Dynamic and Evolving Graphs*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 189–200. [Online]. Available: http://dx.doi.org/10.1007/11618058_18

[21] Y. Frishman and A. Tal, "Dynamic drawing of clustered graphs," in *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*. IEEE, 2004, pp. 191–198.

[22] M. Itoh, M. Toyoda, and M. Kitsuregawa, "An interactive visualization framework for time-series of web graphs in a 3d environment," in *2010 14th International Conference Information Visualisation*. IEEE, 2010, pp. 54–60.

[23] C. Collberg, S. Kobourov, J. Nagra, J. Pitts, and K. Wampler, "A system for graph-based visualization of the evolution of software," in *Proceedings of the 2003 ACM Symposium on Software Visualization*, ser. SoftVis '03, 2003.

[24] K. Boitmanis, U. Brandes, and C. Pich, "Visualizing internet evolution on the autonomous systems level," in *International Symposium on Graph Drawing*. Springer, 2007, pp. 365–376.

[25] K. Andrews, M. Wohlfahrt, and G. Wurzinger, "Visual graph comparison," in *2009 13th International Conference Information Visualisation*. IEEE, 2009, pp. 62–67.

[26] K. M. Hall, "An r-dimensional quadratic placement algorithm," *Management science*, vol. 17, no. 3, pp. 219–229, 1970.

[27] I. Borg and P. J. Groenen, *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

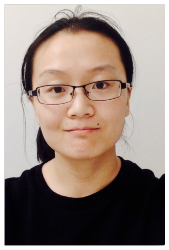
[28] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The princeton shape benchmark," in *Shape modeling applications, 2004. Proceedings*. IEEE, 2004, pp. 167–178.

[29] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "ShapeNet: An Information-Rich 3D Model Repository," Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.

[30] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas, "A scalable active framework for region annotation in 3d shape collections," *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, 2016.

[31] M. Fisher, D. Ritchie, M. Savva, T. Funkhouser, and P. Hanrahan, "Example-based synthesis of 3d object arrangements," *ACM Trans. on Graphics*, vol. 31, no. 6, p. 135, 2012.

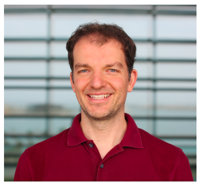
[32] Y.-Y. Ahn, S. E. Ahnert, J. P. Bagrow, and A.-L. Barabási, "Flavor network and the principles of food pairing," *Scientific reports*, vol. 1, 2011.



Jing Ren Jing Ren is currently a PhD student in Computer Science at King Abdullah University of Science and Technology (KAUST) supervised by Prof. Peter Wonka. She received the BSc degree in Mathematics and Applied Mathematics from Zhejiang University, China, in 2014, the MSc degree in Mathematical and Computational Finance from the University of Oxford, UK, in 2015.

Jens Schneider Jens Schneider is a staff scientist at the Visual Computing Center at King Abdullah University of Science and Technology. He received a diploma in Computer Science from RWTH Aachen, Germany and a doctorate from Technische Universität München, Germany. His research interest includes interactive and GPU-based visualization, GPU-friendly data compression and data structures, large-scale visualization, and level-of-detail algorithms.

Maks Ovsjanikov Maks Ovsjanikov is an Assistant Professor at Ecole Polytechnique with a CNRS chaire d'excellence and the Jean Marjoulet professorial chair. He received an Excellence in Research Award from the Institute for Computational and Mathematical Engineering at Stanford University for his work on spectral methods in shape analysis, and the Eurographics Young Researcher Award in 2014 "In recognition of his outstanding contributions to theoretical foundations of non-rigid shape matching". He has served on the technical program committees of various international conferences, is a member of the editorial board of Computer Graphics Forum and has co-chaired the Symposium on Geometry Processing in 2016.



Peter Wonka Peter Wonka is Associate Director of the Visual Computing center (VCC) at King Abdullah University of Science and Technology (KAUST) and Professor in the Computer Science program. Peter Wonka received his doctorate from the Technical University of Vienna in computer science. Additionally, he received a Masters of Science in Urban Planning from the same institution. After his PhD, Dr. Wonka worked as postdoctoral researcher at the Georgia Institute of Technology and as faculty at Arizona State University. His research interests include various topics in computer graphics, visualization, remote sensing, computer vision, image processing, machine learning, and data mining. He currently serves as Associate Editor for ACM Transactions on Graphics, IEEE Computer Graphics and Applications, and IEEE Transactions on Graphics and Visualization.

Appendix A
Proof of Proposition 1

Proof. Recall that we denote $B_{pq} = \frac{1}{2}\mu_{pq}S_{pq}$ and $D_{pq} = \frac{1}{2}\mu_{pq}T_{pq}$. According to the construction of the correspondence matrix S and T, we have $S_{pq} = T_{qp}$ and $T_{pq} = S_{qp}$. Therefore, with the assumption $\mu_{pq} = \mu_{qp}$ we have

$$\begin{aligned}E_2(X) &= \sum_{1\le p<q\le n} \|\mu_{pq}\big(S_{pq}X^{(p)} - T_{pq}X^{(q)}\big)\|_F^2.\\ &= \frac{1}{2}\sum_{p=1}^{n}\sum_{q\neq p} \|\mu_{pq}\big(S_{pq}X^{(p)} - T_{pq}X^{(q)}\big)\|_F^2.\\ &= \sum_{p=1}^{n}\sum_{q\neq p}\|B_{pq}X^{(p)} - D_{pq}X^{(q)}\|_F^2.\end{aligned}$$

For $\forall y \in \mathbb{R}^m$, where $y = (y_1^T\ y_2^T\ \dots\ y_n^T)^T, y_k \in \mathbb{R}^{m_k}, m = \sum_{k=1}^{n} m_k$, we have:

$$\begin{aligned}y^TWy =& \sum_{p=1}^{n} y_p^T W_{pp} y_p + \sum_{p=1}^{n}\sum_{q\neq p} y_p^T W_{pq} y_q\\ =& \sum_{p=1}^{n} y_p^T (L^p)^T L^p y_p +\\ & \sum_{p=1}^{n} y_p^T \Big(\sum_{k\neq p} (B_{pk}^T B_{pk} + D_{kp}^T D_{kp})\Big) y_p +\\ & \sum_{p=1}^{n}\sum_{q\neq p} y_p^T\Big(-\big(B_{pq}^T D_{pq} + D_{qp}^T B_{qp}\big)\Big) y_q\\ =& \sum_{p=1}^{n} y_p^T (L^p)^T L^p y_p + \sum_{p=1}^{n}\sum_{q\neq p} y_p^T B_{pq}^T B_{pq} y_p +\\ & \sum_{q=1}^{n}\sum_{p\neq q} y_q^T D_{pq}^T D_{pq} y_q -\\ & \sum_{p=1}^{n}\sum_{q\neq p} y_p^T B_{pq}^T D_{pq} y_q - \sum_{q=1}^{n}\sum_{p\neq q} y_q^T D_{pq}^T B_{pq} y_p\\ =& \sum_{p=1}^{n} y_p^T (L^p)^T L^p y_p + \sum_{p=1}^{n}\sum_{q\neq p}\Big(y_p^T B_{pq}^T B_{pq} y_p +\\ & y_q^T D_{pq}^T D_{pq} y_q - y_p^T B_{pq}^T D_{pq} y_q - y_q^T D_{pq}^T B_{pq} y_p\Big)\\ =& \sum_{p=1}^{n} y_p^T (L^p)^T L^p y_p +\\ & \sum_{p=1}^{n}\sum_{q\neq p}\Big(B_{pq}y_p - D_{pq}y_q\Big)^T\Big(B_{pq}y_p - D_{pq}y_q\Big)\\ =& \sum_{k=1}^{n}\|L^k y_k\|_F^2 + \sum_{p=1}^{n}\sum_{q\neq p}\|B_{pq}y_p - D_{pq}y_q\|_F^2\\ =& E_1(y) + E_2(y)\end{aligned}$$

Thus, we have the Rayleigh quotient $\frac{y^TWy}{y^Ty} = \frac{E_1(y)+E_2(y)}{\|y\|^2}$, and the result follows by the Min-max theorem. □

Appendix B
Proof of Proposition 2

By construction of $g(X, Z)$:

$$\begin{aligned}F(X) =& \sum_{k=1}^{n}\|L^kX^{(k)}\|_F^2 + \sum_{p=1}^{n}\sum_{q\neq p}\|B_{pq}X^{(p)} - D_{pq}X^{(q)}\|_F^2\\ &+ \sum_{k=1}^{n}\sum_{1\le i<j\le m_k}\lambda_{ij}^{(k)}\Big(\|X_i^{(k)} - X_j^{(k)}\| - \delta_{ij}^{(k)}\Big)^2\\ =& \sum_{k=1}^{n}\Bigg(\|L^kX^{(k)}\|_F^2 + \sum_{p\neq k}\frac{1}{2}\Big(\|B_{pk}X^{(p)} - D_{pk}X^{(k)}\|_F^2\\ & + \|B_{kp}X^{(k)} - D_{kp}X^{(p)}\|_F^2\Big)\\ & + \sum_{1\le i<j\le m_k}\lambda_{ij}^{(k)}\Big(\|X_i^{(k)} - X_j^{(k)}\| - \delta_{ij}^{(k)}\Big)^2\Bigg)\end{aligned}$$

The k-th term of $F(X)$:

$$\begin{aligned}&\|L^kX^{(k)}\|_F^2 = \mathrm{tr}\Big(X^{(k)T}(L^k)^TL^kX^{(k)}\Big)\\ &\|B_{pk}X^{(p)} - D_{pk}X^{(k)}\|_F^2 = \mathrm{tr}\Big(X^{(p)T}B_{pk}^TB_{pk}X^{(p)} -\\ &\quad 2X^{(p)T}B_{pk}^TD_{pk}X^{(k)} + X^{(k)T}D_{pk}^TD_{pk}X^{(k)}\Big)\\ &\|B_{kp}X^{(k)} - D_{kp}X^{(p)}\|_F^2 = \mathrm{tr}\Big(X^{(k)T}B_{kp}^TB_{kp}X^{(k)} -\\ &\quad 2X^{(p)T}D_{kp}^TB_{kp}X^{(k)} + X^{(p)T}D_{kp}^TD_{kp}X^{(p)}\Big)\\ &\sum_{1\le i<j\le m_k}\lambda_{ij}^{(k)}\Big(\|X_i^{(k)} - X_j^{(k)}\| - \delta_{ij}^{(k)}\Big)^2 =\\ &\mathrm{tr}\Big(X^{(k)T}L_\lambda^kX^{(k)}\Big) + c_k -\\ &2\sum_{1\le i<j\le m_k}\lambda_{ij}^{(k)}\delta_{ij}^{(k)}\|X_i^{(k)} - X_j^{(k)}\|\end{aligned}$$

where $c_k = \sum_{1\le i<j\le m_k}\lambda_{ij}^{(k)}\delta_{ij}^{(k)2}$ is a constant.Therefore, the energy function could be reformulated as:

$$\begin{aligned}&F(X) =\\ &\sum_{k=1}^{n}\Bigg(\mathrm{tr}\Big(X^{(k)T}M_kX^{(k)}\Big) + \sum_{p\neq k}\mathrm{tr}\Big(X^{(p)T}N_{pk}X^{(k)}\Big) +\\ &\sum_{p\neq k}\mathrm{tr}\Big(X^{(p)T}Q_{pk}X^{(p)}\Big) + c_k -\\ &2\sum_{1\le i<j\le m_k}\lambda_{ij}^{(k)}\delta_{ij}^{(k)}\|X_i^{(k)} - X_j^{(k)}\|\Bigg)\end{aligned}$$

where

$$\begin{aligned}M_k &= (L^k)^TL^k + \frac{1}{2}\sum_{p\neq k}\big(D_{pk}^TD_{pk} + B_{kp}^TB_{kp}\big) + L_\lambda^k\\ N_{pk} &= -B_{pk}^TD_{pk} - D_{kp}^TB_{kp}\\ Q_{pk} &= \frac{1}{2}\big(B_{pk}^TB_{pk} + D_{kp}^TD_{kp}\big)\\ L_\lambda^k &= \sum_{1\le i<j\le m_k}\lambda_{ij}^{(k)}(e_i - e_j)(e_i - e_j)^T\end{aligned}$$

where for a given k, $\{e_i\}_{i=1}^{m_k}$ are the standard basis for $\mathbb{R}^{m_k}$.

To construct the majorizing function for $F(X)$, introducing the variables $Z^{(k)}, k = 1, 2, ..., n$, we have (according to Cauchy-Schwartz Inequality):

$$\langle X_i^{(k)} - X_j^{(k)}, Z_i^{(k)} - Z_j^{(k)} \rangle \leq \|X_i^{(k)} - X_j^{(k)}\| \|Z_i^{(k)} - Z_j^{(k)}\|$$

The equality holds when $Z = X$. Therefore:

$$\begin{aligned} -\|X_i^{(k)} - X_j^{(k)}\| &\leq -\frac{\langle X_i^{(k)} - X_j^{(k)}, Z_i^{(k)} - Z_j^{(k)} \rangle}{\|Z_i^{(k)} - Z_j^{(k)}\|} \\ &= -\frac{\text{tr}\big(X^{(k)T}(e_i - e_j)(e_i - e_j)^T Z^{(k)}\big)}{\|Z_i^{(k)} - Z_j^{(k)}\|} \end{aligned}$$

$$\begin{aligned} F(X) \leq \sum_{k=1}^{n} \Bigg(&\text{tr}\Big(X^{(k)T} M_k X^{(k)}\Big) + \sum_{p \neq k} \text{tr}\Big(X^{(p)T} N_{pk} X^{(k)}\Big) + \\ &\sum_{p \neq k} \text{tr}\Big(X^{(p)T} Q_{pk} X^{(p)}\Big) + c_k - \\ &2 \sum_{1 \leq i < j \leq m_k} \lambda_{ij}^{(k)} \delta_{ij}^{(k)} \frac{\text{tr}\big(X^{(k)T}(e_i - e_j)(e_i - e_j)^T Z^{(k)}\big)}{\|Z_i^{(k)} - Z_j^{(k)}\|} \Bigg) \end{aligned}$$

Therefore, we can define the majorizing function as follows:

$$\begin{aligned} &g(X, Z) := \\ &\sum_{k=1}^{n} \Bigg(\text{tr}\Big(X^{(k)T} M_k X^{(k)}\Big) + \sum_{p \neq k} \text{tr}\Big(X^{(p)T} N_{pk} X^{(k)}\Big) \\ &+ \sum_{p \neq k} \text{tr}\Big(X^{(p)T} Q_{pk} X^{(p)}\Big) + c_k - \text{tr}\Big(X^{(k)T} W^{Z^{(k)}} Z^{(k)}\Big) \Bigg) \end{aligned}$$

where $W^{Z^{(k)}} = \sum_{1 \leq i < j \leq m_k} \frac{2\lambda_{ij}^{(k)} \delta_{ij}^{(k)} (e_i - e_j)(e_i - e_j)^T}{\|Z_i^{(k)} - Z_j^{(k)}\|}$

According to the construction process, we have $g(X, Z) \geq F(X)$, and the equality holds when $Z = X$. Moreover, $g(X, Z)$ is quadratic in X given any fixed Z. M_k is a summation of a set of positive semi-definite matrices, therefore $\nabla_X g(X, Z) = 0$ gives a descent direction. Therefore, we could choose X^u satisfying $\nabla_X g(X^u, Z) = 0$ for the second step of majorization algorithm.

$$\begin{aligned} \frac{\partial g(X, Z)}{\partial X^{(j)}} = &\Big(M_j + M_j^T + \sum_{k=1, k \neq j}^{n} (Q_{jk} + Q_{jk}^T)\Big) X^{(j)} + \\ &\sum_{p=1, p \neq j}^{n} (N_{pj}^T + N_{jp}) X^{(p)} - W^{Z^{(j)}} Z^{(j)} \end{aligned}$$

Therefore, if Z is fixed, we have $\nabla_X g(X, Z) = VX - UZ$.

where $V, U \in \mathbb{R}^{m \times m}, m = \sum_{i=k}^{n} m_k$. V is defined by blocks of $V_{pq} \in \mathbb{R}^{m_p \times m_q}$ as follows:

$$V_{pq} = \begin{cases} M_p + M_p^T + \sum_{k \neq p} \big(Q_{pk} + Q_{pk}^T\big) & \text{if } p = q \\ N_{pq} + N_{qp}^T & \text{otherwise} \end{cases}$$

and $U = \text{diag}\big(W^{Z^{(1)}}, W^{Z^{(2)}}, ..., W^{Z^{(n)}}\big)$

Therefore, we could find X^u by solving the following linear system with a fixed Z:

$$VX^u = U(Z)Z$$