

# Same Graph, Different Views (3D Multiview Graph Visualization)

Iqbal Hossain, Vahan Huroyan, Stephen Kobourov, Raymundo Navarrete

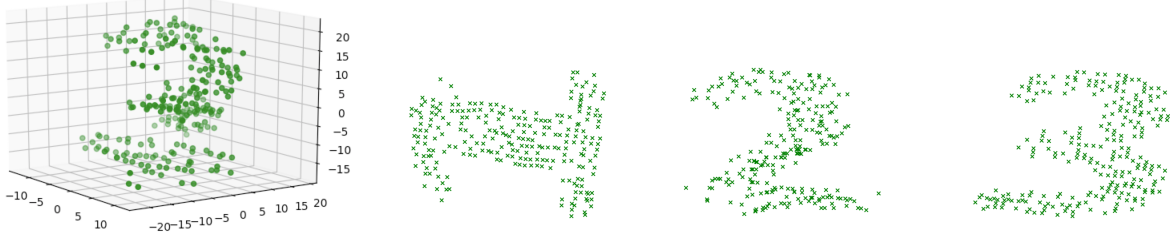


Fig. 1: Given several distances between a set of objects and matching projection planes, 3DMGV computes 3D coordinates for all the objects (left image), such that when projected to the corresponding 2D planes the distances in each plane match the corresponding input distances (the right three images). The input dataset is inspired by a sculpture “1, 2, 3” by James Hopkins.

**Abstract**— We describe a method for simultaneous visualization of multiple pairwise distances in 3 dimensional (3D) space. Traditionally, when visualizing a relational dataset, such as a graph made up of a set of objects and relations between them, we strive for a drawing in the 2D plane that matches the Euclidean distances in the plane to the graph distances (e.g., determined by all-pairs shortest paths). If we want to visualize two or more relationships (sets of edges) on the same set of objects (nodes) in 2D, we can try to optimize the layout from the point of view of the union graph, but this comes at the expense of high distortion. We propose an approach that uses 3D to place the nodes of the graph, along with projections (planes) that show each of the given relationships. Our 3D Multiview Graph Visualization (3DMGV) method is based on non-linear dimensionality reduction that generalizes multidimensional scaling. We consider two versions of the problem: in the first one we are given the input graphs and the projections (e.g., if we have 3 different relationships we can use the three orthogonal directions of the unit cube). In the second version of the problem we also compute the best projections as part of the optimization. We experimentally evaluate 3DMGV using synthetic datasets, that illustrate the quality of the resulting solutions. Finally, we provide a functional prototype which implements both settings.

**Index Terms**—Graph visualization, Dimensionality reduction, Multidimensional scaling, Mental map preservation.

## 1 Introduction

Typically, when given a graph  $G = (V, E)$  the goal of visualizing it can be summed up as finding some 2D layout that represents the underlying structure. Similarly, given some high dimensional dataset, the goal of visualizing it often is interpreted as finding some 2D placement of the objects such that similar objects are close to each other and dissimilar ones are far. Dating back to the 1960s, a classical tool that is used for both graphs and high dimensional dataset visualization, is Multidimensional Scaling (MDS) which aims to preserve the distances between all pairs of nodes/objects [31]. More recent dimensionality reduction approaches, such as t-SNE [26], aim to preserve local neighborhoods and clustering.

Now consider a more general case when the input is a set of vertices  $V$  (e.g., researchers in one university) and several rela-

tionships defined between them  $E_1, E_2, E_3$  (e.g., joint research publications, joint research proposals, membership in different department). We would like to compute a layout  $L$  in 3D as well as 3 planes such that when  $L$  is projected onto plane  $P_1$  we see the graph  $G = (V, E_1)$  so that distances between vertices in the plane  $P_1$  correspond to the distances defined by  $E_1$ . Similarly, when  $L$  is projected onto plane  $P_2$  we see the graph  $G = (V, E_2)$  so that distances between vertices in the plane  $P_2$  correspond to the distances defined by  $E_2$ , and the same for  $P_3$  and  $E_3$ .

In the high dimensional setting, the goal is to perform dimensionality reduction to 3D, given a set of objects, several pairwise distance functions between them, and the same number of projection planes. The optimization goal now is to simultaneously preserve the distances between the objects when projected to the corresponding planes.

In both settings, this is a strict generalization of the underlying classical problem, which can be seen as a special case when only one pairwise distance function is given. Even this special case is known to be difficult as the standard optimization approaches such as gradient descent do not necessarily converge to the global optimum. Nevertheless, in practice, when there is clear structure in the given graph, MDS is often likely to find a good local optimum and as we show in this paper, the simultaneous optimization of our 3DMGV produces good solutions.

We describe the 3DMGV method in detail and also briefly mention how it is implemented. We consider two different settings: one where the projection planes are given as part of the input (e.g., the three sides of a 3D cube) and the second

- Iqbal Hossain and Stephen Kobourov are with the Computer Science Department of The University of Arizona, E-mails: [hossain@email.arizona.edu](mailto:hossain@email.arizona.edu) and [kobourov@cs.arizona.edu](mailto:kobourov@cs.arizona.edu).
- Vahan Huroyan and Raymundo Navarrete are with the Department of Mathematics of The University of Arizona, E-mails: [vahanhuroyan@math.arizona.edu](mailto:vahanhuroyan@math.arizona.edu) and [aymundo@math.arizona.edu](mailto:aymundo@math.arizona.edu).

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: [reprints@ieee.org](mailto:reprints@ieee.org). Digital Object Identifier: [xx.xxx/TVCG.201x.xxxxxx](https://doi.org/10.1109/TVCG.201x.xxxxxx)

where computing the projection planes is part of the optimization. Both settings have been implemented and work well in practice. We illustrate performance with several examples.

A common approach for visualizing different relationships on the same set of objects involve small multiples and often some mechanism (such as brushing and linking) to connect the same objects in the different views, or morphing from one view to the other. In contrast, 3DMGV produces one 3D layout and each of the different views is a 2D projection. In this way, 3DMGV attempts to balance the two main desirable qualities of good visualization of multiple relationships defined on the same set of data: the *readability* of each individual view (typically captured by a faithful embedding in 2D) and *mental map preservation* (typically captured by keeping the objects in the same position across different views). This cannot be accomplished effectively in 2D as there simply is not enough space to realize more than one relationship well. This becomes more plausible in 3D, and with the advent of virtual reality and augmented reality systems, 3D visualization and interaction itself is becoming a reality. Still, when presenting 3D results in a paper we are limited to showing 2D snapshots. We include 3D visualizations with interactive examples on this webpage <https://uamap-dev.arl.arizona.edu/static/3DMGV/index.html>

### 1.1 Previous work

We review work on visualizing multivariate and multilayer networks, network layout algorithms, multidimensional scaling, simultaneous embedding, and 3D reconstruction.

**Multivariate network visualization.** Multivariate [19] and multilayer [16] graph visualization has received a great deal of attention in the last couple of decades. Multi-label, multi-edge, multi-relational, multiplex, multi-modal and many other variants are cleverly encapsulated by the general multilayer network definition of Kivelä et al. [20].

Wattenberg’s PivotGraph [36] system can visualize and analyze multivariate graphs not using a global graph layout but rather a grid-based approach focusing on different relationship between node attributes. Semantic substrates [32] unfold multiple attributes of a graph, a pair of attributes at a time, using two dimensions. Pretorius and van Wijk [29] describe an interactive system that relies on clustering of both nodes and edges and interactive exploration using brushing and linking (as well as parallel histograms) to show different graph attributes. *GraphDice* [5] is an interactive multivariate graph visualization system that allows the selection of attributes from an overview plot matrix. This results in a cross dimensional node-link plot for every combination of attributes arranged as a matrix. When different attributes are selected, the matrix of node-link diagrams morphs from the old to the new. This system is built on the earlier *ScatterDice* system [13].

Different from our approach, most of the earlier methods focus on interactive visualizations of multivariate graphs where changing queries result in changing layouts and views. The idea behind our 3DMGV approach is to produce one 3D layout of the input graph, and several projection planes, such that each attribute corresponds to a projection plane in which geometric distances correspond to the graph distances specified by the particular attribute. The main advantage of this approach is that it should help preserve the viewer’s 3D mental map, while also capturing different relationships in different projections of the same underlying layout.

**Network layout algorithms.** Most basic network layouts are obtained using force-directed algorithms. Also known as spring embedders, such algorithms calculate the layout of the underlying graph using only information contained within the structure of the graph itself, rather than relying on domain-specific knowledge [21]. Visual analytics systems for graphs usually focus on interaction [34]. MDS-like approaches to drawing graphs are exemplified in algorithms such as that of

Kamada-Kawai [18], Koren and Carmel [23]. Most commonly used graph drawing systems, such as Graphviz [12], Pajek [4], Tulip [2] and Gephi [3], provide options to visualize graphs in 3D based on MDS-like optimization. Variants of MDS are used in many graph layout systems, including [9, 14, 28, 35]. Other approaches to exploring layouts in 3D include 3D hyperbolic and spherical spaces [10, 22, 27].

**Multidimensional scaling.** Multidimensional scaling (MDS) is a well known dimensionality reduction and data visualization technique. The problem was first studied in the non-metric setting by Shepard [31] and Kruskal [25]. Non-metric MDS recovers structure from measures of similarity, based on the assumption of a reproducible ordering between the distances, rather than relying on the exact distances. The metric variant of MDS is more frequently used and it relies on the exact distances. The goal of metric MDS is to place objects in some low dimensional space so as to preserve the given pairwise distances between the objects. Given a distance matrix (pairwise dissimilarity matrix)  $D = (d_{ij})_{i,j=1}^{n,n}$ , between  $n$  objects, the objective function for MDS is

$$S(x_1, \dots, x_n) = \sum_{i>j} (d_{ij} - \|x_i - x_j\|)^2. \quad (1)$$

The function defined in (1) is called the stress function. Some well known techniques for minimizing the stress function (1) are standard gradient descent, stochastic gradient descent [7], and stress majorization [14].

**Simultaneous embedding.** This problem is also related to simultaneous graph embedding and matched drawings of graphs [6]. Specifically, in simultaneous geometric embedding of two or more planar graphs requires planar straight-line drawings of each of the graphs, such that common vertices have the same 2D coordinates in all drawings. This setting is very restrictive and solutions are guaranteed to exist for very restricted type of input graphs, such as two paths [8], while instances with no solutions can be constructed from a pair of trees [15] or even a (path, tree) pair [1]. Matched drawings require straight-line drawings of the two or more input graph such that each common vertex has the same  $y$ -coordinate in all drawings. Pairs of trees and triples of cycles always have a matched drawing [17]. In general, instances with no solution can be constructed from a pair of planar graphs, or even a (planar graph, tree) pair [11]. Note that matched pairs of drawings can be obtained from the 3DMGV embedding for every pair of graphs using the intersection line between the corresponding pairs of projection planes as the shared  $y$ -coordinate in the pair of matched drawings.

**3D Reconstruction.** Our problem is also related a 3D reconstruction problem from a collection of 2D images. This problem has been widely studied in different settings, including reconstructing the underlying real 3D structure from large collections of 2D photos [33]. More restricted variants are even closer to our setting [24, 30]. Note however, that in our problem we have a constant number of inputs (distance matrices or graphs) and the projections we anticipate can be fixed or computed as a part of the optimization.

### 1.2 Our Contribution

The main contribution in this paper is a generalization of MDS to multiple distance matrices. This is at the core of the proposed 3DMGV method for visualizing the same dataset/graph in 3D from several different views, each of which captures a different set of distances/relationships. We consider two main variants: one in which each of the different distances/relationships is associated with a specific 2D projection plane, and the other where computing the projection planes is also part of the optimization.

## 2 Multiview Graph Visualization

We begin this section with a brief review of the standard MDS problem and then an overview of our multiview-MDS formulation. Let  $d$  be an  $n$  by  $n$  matrix containing pairwise dissimilarity measures between  $n$  objects. The goal in multidimensional analysis is to assign positions  $x_1, x_2, \dots, x_n \in \mathbb{R}^p$  to the  $n$  objects so that the resulting pairwise distances  $\|x_i - x_j\|$  are as close as possible to the observed pairwise dissimilarities  $d_{ij}$ , as measured by the MDS stress function (1). If the minimum of the MDS stress function is zero, then the objects can be positioned so that their pairwise distances exactly represent the pairwise dissimilarities  $d$ . If the minimum of the MDS stress function is greater than zero, a minimizer of (1) still provides an approximate way to visualize the dissimilarities.

Suppose that instead of a single pairwise dissimilarity matrix  $D$ , we observe multiple pairwise dissimilarities matrices  $D^1, D^2, \dots, D^l$  for the same set of  $n$  objects. It is natural to ask if an embedding  $x_1, x_2, \dots, x_n \in \mathbb{R}^p$  of the objects exists so that the different dissimilarities can be visualized by the relative positions of  $x_1, x_2, \dots, x_n$ , but there is no way to accomplish this without further assumptions about the relationship between the different pairwise dissimilarity matrices.

Motivated by the problem of 3D reconstruction from multiple 2D images, we consider the following question: is it possible to place the  $n$  objects under consideration in 3D,  $x_1, x_2, \dots, x_n \in \mathbb{R}^3$ , so that the different pairwise dissimilarity matrices are equal to the pairwise distances between the objects after being projected to different 2-dimensional subspaces of  $\mathbb{R}^3$ ? For example, the pairwise dissimilarity matrices  $D^1, D^2$ , and  $D^3$  can be given by  $D_{ij}^l = \|E_l x_i - E_l x_j\|$ , where  $1 \leq l \leq 3$  and  $E_1, E_2$  and  $E_3$  are the 3 by 3 orthogonal projection matrices that project onto the  $xy$ ,  $xz$ , and  $yz$  coordinate planes. In this scenario, we can ask if the positions  $x_1, x_2, \dots, x_n$  can be recovered from the pairwise dissimilarity matrices, assuming that the corresponding projections are known, or if both the positions and projections can be recovered from the pairwise dissimilarity matrices alone. Even for sets of pairwise dissimilarity measures that are not generated this way, this assumptions can be used to form visualizations that can simultaneously illustrate the different dissimilarity measures.

### 2.1 3DMGV with Fixed Projections

We consider the following problem. We have  $n$  objects that we wish to embed in 3D space. We begin with three pairwise dissimilarity matrices  $D^1, D^2$  and  $D^3$ , along with the corresponding orthogonal projection matrices  $\Pi_1, \Pi_2$  and  $\Pi_3$ . We wish to find the positions  $x_1, x_2, \dots, x_n \in \mathbb{R}^3$  that best agree with the set of distance/projection pairs. For this purpose, we define the multiview-MDS stress function:

$$S_M(x_1, x_2, \dots, x_n) = \sum_{l=1}^3 \sum_{i>j} \left( D_{ij}^l - \|\Pi_l x_i - \Pi_l x_j\| \right)^2 \quad (2)$$

This function measures disagreement between each of the pairwise dissimilarity matrices and the pairwise distances between the corresponding projected positions. The goal is to find a set of positions that minimize the multiview-MDS stress function (2), that is

$$\text{minimize } S_M(x_1, x_2, \dots, x_n) \quad (3)$$

In our discussion and experiment, the matrices  $\Pi_1, \Pi_2$  and  $\Pi_3$  are always orthogonal projection matrices or rank 2. A  $3 \times 3$  matrix  $\Pi$  is an orthogonal projection matrix if  $\Pi^2 = \Pi = \Pi^T$ . For example, a natural triple is to fix the projection matrices as the projections onto the  $xy$ ,  $xz$  and  $yz$  planes, as given by

$$E_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, E_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, E_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (4)$$

We will refer to these as the “unit cube projections”. Another set of fixed projection matrices that we use in our experiments are projections onto the  $xz$  plane and the two planes that form  $\pi/3$  and  $2\pi/3$  angles with  $xz$  plane (corresponding to a rotation of the 3D input around the  $z$ -axis):

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} \frac{1}{4} & \frac{\sqrt{3}}{4} & 0 \\ \frac{\sqrt{3}}{4} & \frac{3}{4} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} \frac{1}{4} & \frac{-\sqrt{3}}{4} & 0 \\ \frac{-\sqrt{3}}{4} & \frac{3}{4} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5)$$

Problem (3) can be formulated for any number of distance matrices and corresponding projections, and the dimension of the ambient space and rank of the projections can be arbitrary. The restrictions on  $\Pi_1, \Pi_2$  and  $\Pi_3$  could also be removed to allow for more general linear or even non-linear maps. For clarity, we limit our presentation to the setup with 3 distance matrices and 3 orthogonal projections.

An important property of the objective function for multiview MDS in (1) is that it is differentiable. We solve (3) with gradient descent. Since the objective function defined in (2) is not convex, one needs to be careful when choosing the initial configuration and learning rate (the size of the movement along the gradient) in order for the algorithm to converge to a local minimum. Thus, to find the optimal point an appropriate initialization and learning rate are required. We summarize the gradient descent algorithm for the multiview MDS in Algorithm 1.

---

#### Algorithm 1 Gradient Descent Algorithm for 3DMGV

---

**Input:** Distance matrices:  $D^1, D^2, D^3 \in \mathbb{R}^{n \times n}$ , learning rate:  $\mu$ , maximum number of iterations:  $N$   
Initialize  $X^{(0)} = [x_1^0, x_2^0, \dots, x_n^0] \in \mathbb{R}^{3 \times n}$   
**while**  $i \leq N$  **do**  
     $X^{(i+1)} = X^{(i)} - \mu \nabla_X S_M(X^{(i)})$   
**end while**  
**Output:**  $x_1^N, x_2^N, \dots, x_n^N \in \mathbb{R}^3$

---

### 3 3DMGV with Varying Projections

In this section, we again consider minimization of the multiview-MDS stress function (2), but we no longer assume that the projection matrices are given. Our goal is then to find both the positions  $x_1, x_2, \dots, x_n \in \mathbb{R}^3$  and the projection matrices  $\Pi_1, \Pi_2$  and  $\Pi_3$  that best capture the given distance matrices  $D^1, D^2$  and  $D^3$ .

We now formulate the 3DMGV problem for varying projections. As before, we assume that  $D^1, D^2$  and  $D^3$  are the distance matrices observed after projecting by  $\Pi_1, \Pi_2$  and  $\Pi_3$ , respectively, but these projection matrices are no longer known. Note that a  $3 \times 3$  matrix  $\Pi$  is a rank-2 orthogonal projection matrix if and only if  $\Pi = QQ^T$ , where  $Q$  is an orthonormal  $3 \times 2$  matrix. Let  $\mathbb{O}^{3 \times 2}$  be the set of all orthogonal  $3 \times 2$  matrices and let  $X = [x_1, x_2, \dots, x_n]$ . The multiview MDS stress function (2) takes the form

$$S_M(X; Q_1, Q_2, Q_3) = \sum_{l=1}^3 \sum_{i>j} \left( D_{ij}^l - \|Q_l Q_l^T (x_i - x_j)\| \right)^2 \quad (6)$$

and the optimization problem becomes

$$\begin{aligned} &\text{minimize } S_M(X; Q_1, Q_2, Q_3) \\ &\text{subject to } Q_1, Q_2, Q_3 \in \mathbb{O}^{3 \times 2} \end{aligned} \quad (7)$$

Solving problem (7) presents difficulties not found in the classical MDS problem. The multiview MDS stress function (6) is differentiable with respect to both the positions  $X$  and

orthogonal matrices  $Q_l$ , but we have found that simply using gradient descent on the combined variable  $[X, Q_1, Q_2, Q_3]$  does not produce good results, due to the non-convexity of the problem. Instead, we make use of a strategy called projected gradient descent, where the algorithm alternates between minimizing (6) with respect to the matrix  $X$  and each of the projections. At any given iteration, in order to update the positions  $X$ , the current projection matrices are fixed and Algorithm 1 is used. Then, we fix  $X$  and minimize (6) for each of the projection matrices separately. The procedure is repeated until convergence. Since convergence is not guaranteed, the algorithm may be terminated when a fixed number of iterations is reached. This algorithm is summarized in Algorithm 2. Details on each of the steps are presented afterwards.

---

**Algorithm 2** 3DMGV with Unknown Projections

---

**Input:** Distance matrices:  $D^1, D^2, D^3$ , learning rates:  $\mu_X$  and  $\mu_Q$ , number of initial iterations:  $N_X$  and  $N_Q$ , number of loops:  $M$ , number of iterations per loop:  $M_X$  and  $M_Q$ .  
 Compute  $X^{(0)}$  and  $Q_1^{(0)}, Q_2^{(0)}$  and  $Q_3^{(0)}$  using Algorithm 4.  
**while**  $i \leq M$  **do**  
   Compute  $X^{(i+1)}$  using Algorithm 1 with initial positions  $X^{(i)}$  and fixed projections  $Q_1^{(i)}, Q_2^{(i)}$  and  $Q_3^{(i)}$ .  
   For  $l \in \{1, 2, 3\}$ , compute  $Q_l^{(i+1)}$  using Algorithm 3 using initial orthogonal matrix  $Q_l^{(i)}$  and fixed positions  $X^{(i+1)}$ .  
**end while**  
**Output:**  $X^{(M)}$  and  $Q_1^{(M)}, Q_2^{(M)}$  and  $Q_3^{(M)}$ .

---

Since the set  $\mathbb{O}^{3 \times 2}$  of 3 by 2 orthogonal matrices is not a subspace of  $\mathbb{R}^{3 \times 2}$ , minimizing (6) with respect to  $Q_l$  cannot be accomplished via gradient descent. Instead, we make use of projected gradient descent, where  $Q_l$  is updated by first moving towards the direction of steepest descent, and then projecting back onto the set  $\mathbb{O}^{3 \times 2}$ . If  $A \in \mathbb{R}^{3 \times 2}$  matrix, then the projection of  $A$  onto  $\mathbb{O}^{3 \times 2}$  is the matrix  $\mathcal{P}(A) \in \mathbb{O}^{3 \times 2}$  that minimizes  $\|A - Q\|_F$  among all  $Q \in \mathbb{O}^{3 \times 2}$ . There is a simple way to compute  $\mathcal{P}(A)$ : if  $U\Sigma V^T$  is the reduced singular value decomposition of  $A$ , then  $\mathcal{P}(A) = UV^T$ . The algorithm is summarized in Algorithm 3.

---

**Algorithm 3** Projected gradient descent for 3DMGV

---

**Input:** Distance matrix:  $D$ , fixed positions:  $X$ , initial orthogonal matrix:  $Q^{(0)}$ , learning rate:  $\mu$ , number of iterations:  $N$ .  
**while**  $i \leq N$  **do**  
    $\tilde{Q}^{(i+1)} = Q^{(i)} - \mu \nabla_Q S_M(X; Q^{(i)})$   
    $Q^{(i+1)} = \mathcal{P}(\tilde{Q}^{(i+1)})$   
**end while**  
**Output:**  $Q^{(N)}$

---

In our experiments, we found that the choice of initial positions  $X^{(0)}$  and projections  $\Pi_1^{(0)}, \Pi_2^{(0)}, \Pi_3^{(0)}$  is very important in avoiding convergence to non-optimal local minima (such as one in which all projection planes are parallel to each other). With this in mind, we use some non-random initial plane configurations. We found that a good choice for  $X^{(0)}$  is the one obtained by first assuming that distance matrices  $D^1, D^2, D^3$  can be generated exactly from the unknown  $X^{(0)}$  and the unit cube projection matrices  $E_1, E_2, E_3$  described in Section 2.1. Under these assumptions, the distance matrix  $D \in \mathbb{R}^{n \times n}$  of the positions  $X^{(0)}$  is given exactly by

$$D_{ij} = \sqrt{\frac{((D_{ij}^1)^2 + (D_{ij}^2)^2 + (D_{ij}^3)^2)}{2}}. \quad (8)$$



Fig. 2: One-Two-Three Dataset. The first row contains the original images of digits 1, 2 and 3. The second row shows the sampled 200 points from each digit.

The goal is then to find the set of points  $X^{(0)}$  that minimize the MDS stress function (1) for the computed distance matrix  $D$  using gradient descent. After computing  $X^{(0)}$ , we obtain the initial projection matrices  $\Pi_1^{(0)}, \Pi_2^{(0)}, \Pi_3^{(0)}$  by running Algorithm 3 for each projection, one by one. The initialization algorithm is described in Algorithm 4.

---

**Algorithm 4** 3DMGV Initialization

---

**Input:** Distance matrices:  $D^1, D^2$  and  $D^3$ , learning rates:  $\mu_X$  and  $\mu_Q$ , number of iterations:  $N_X$  and  $N_Q$ .  
 Compute distance matrix  $D$  as shown in 8.  
 Compute  $X^{(0)}$  using gradient descent on the MDS stress function (1) with distance matrix  $D$ , using learning rate  $\mu_X$ , number of iterations  $N_X$ , and random initial conditions.  
 For  $l \in \{1, 2, 3\}$ , compute  $Q_l^{(0)}$  using Algorithm 3 for fixed positions  $X^{(0)}$  and distance matrix  $D^l$ , using learning rate  $\mu_Q$ , number of iterations  $N_Q$ , and random initial conditions.  
**Output:**  $X^{(0)}$  and  $Q_1^{(0)}, Q_2^{(0)}$  and  $Q_3^{(0)}$ .

---

## 4 Experimental Evaluation

In this section we describe the construction of the datasets that we use to evaluate our proposed 3DMGV method in its two settings: with fixed projections and with variable projections. For each dataset, we first describe how it was created and show the outputs of 3DMGV. Note that Algorithm 1, similar to the regular MDS, achieves results up to a rotation and reflection and there is no way to recover the correct positioning of graph/data without any extra information (assuming that the only information available about the dataset are distance matrices).

### 4.1 One-Two-Three Dataset

In the introduction we motivated the 3DMGV problem using the sculpture “1, 2, 3” by James Hopkins (one-two-three statue). To generate the dataset for Fig. 1 we attempted to reverse-engineer the sculpture as follows. We created 2D visualizations of the 3 projections of the statue and fed the 2D distances as input to 3DMGV. To do this, we created black-and-white images of the digits 1, 2 and 3 and sampled equal number of points from each of the 3 images; see Figure 2.

We then use the distances between the points in the 3 projections as input to 3DMGV. We first run Algorithm 1 for only two of the three digits (one-two, one-three and two-three) with 2 fixed projections from (4) and (5); see Figure 3. Note that both Algorithms 1 and 2 return 3D visualizations of the datasets and in Figure 3 we output the corresponding projections of the 3D datasets. We only show the results for (5). By



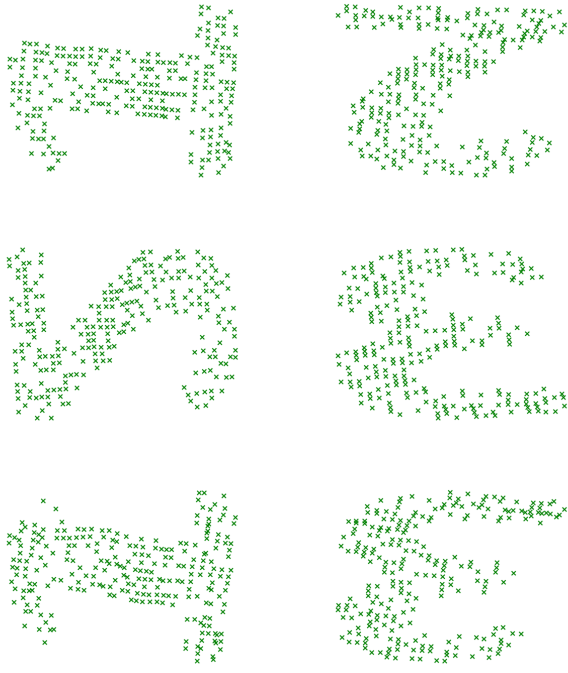


Fig. 3: This figure shows the results of the application of Algorithm 1 to the One-Two-Three dataset using only the first two projections from (5) for different inputs. The first row shows the results for digits 1 and 2, the next row shows the results for digits 2 and 3, and the last row shows the results for digits 1 and 3.

observing all three rows of Figure 3, we conclude that Algorithm 1 with projections from (5) perfectly recovers the shapes of the digits.

Next we applied Algorithm 1 with 2 different sets of projections (from (4) and (5)) to the one-two-three dataset. In Figure 8 we only show the results for the projections from (5). We observe that for projections from (5), the algorithm successfully recovered the shapes of 2 and 3, however the recovered 1 is very similar to 2.

#### 4.2 Circle-Square-Triangle Dataset

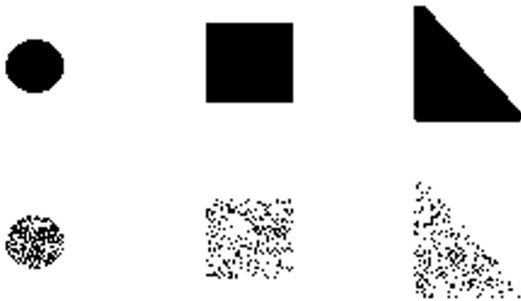


Fig. 4: Circle-Square-Triangle dataset. The first row contains the original images of all shapes. The second row presents the sampled 250 points from each shape.

The next dataset that we create consists of the geometric shapes of a circle, square and triangle. We refer to this dataset

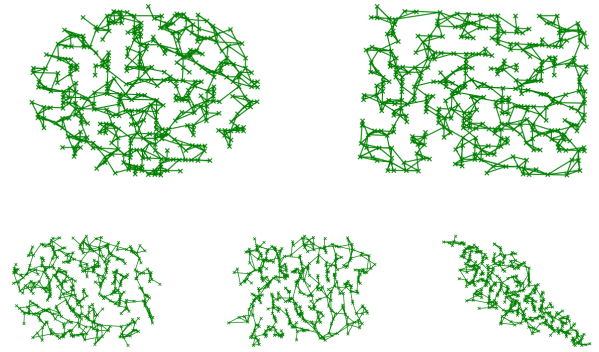


Fig. 5: Demonstration of the application of Algorithm 1 with projections from (4) on the circle-square-triangle dataset. The first row shows the results for the input only circle and square. The second row shows the results of the input circle, square and triangle.

as the circle-square-triangle dataset. Similar to 4.1, we create 3 black and white images with white background, each containing one of the following shapes: circle, square and triangle. We sample equal number of points from each shape, see Figure 4. The goal is to find such a placement of points in 3D such that in fixed projection case the given projections recover the shapes and in the varying projections case the algorithm find the projections for which we recover the 3 shapes.

To make this dataset more challenging we propose to create a graph based on each shape by connecting each point to its  $k$  nearest neighbors. In this case we can create the distance matrix by calculating the all pairs shortest path between each two points.

We present the results of the application of Algorithm 1 on this dataset in Figure 5. For the first row of Figure 5 we input only the datasets for circle and square. We note, that Algorithm 1 captures both shapes very well. The second row of Figure 5 presents the results of Algorithm 1 for the input dataset that contains square, circle and triangle with projections from (4). Note that the shapes are not perfect, but the algorithm recovered the circle in the first figure, square in the second one and triangle in the third one.

#### 4.3 Clusters Dataset Description

One of the many applications of dimensionality reduction is to preprocess the dataset by reducing its dimension and then apply a clustering/classification algorithm. To test whether our proposed algorithm would preserve clusters in a dataset we propose the following setting. Assume we want to visualize data in 3D such that its given 2D projections contain clusters, e.g., see the first row of Figure 6. Each subfigure of the first row of Figure 6 contains 2 clusters, however there is no correspondence between the points, that is, if 2 points are in the same cluster in one of the subfigures, their position in the other ones are arbitrary.

Our goal is to apply Algorithm 1 and 2 and see whether the results preserve the clusters.

The second row of Figure 6 demonstrates the results of the application of Algorithm 2 to the cluster dataset. We note that the cluster information is well preserved in all three projections. We have also applied Algorithm 1 with both projection sets from (4) and (5). However, Alorithm 2 achieved the best results

#### 4.4 Grid-Path graph Dataset Description

The previous three examples were data/shape visualisation examples. However, MDS is also used for graph visualization

update  
value of  
 $k$

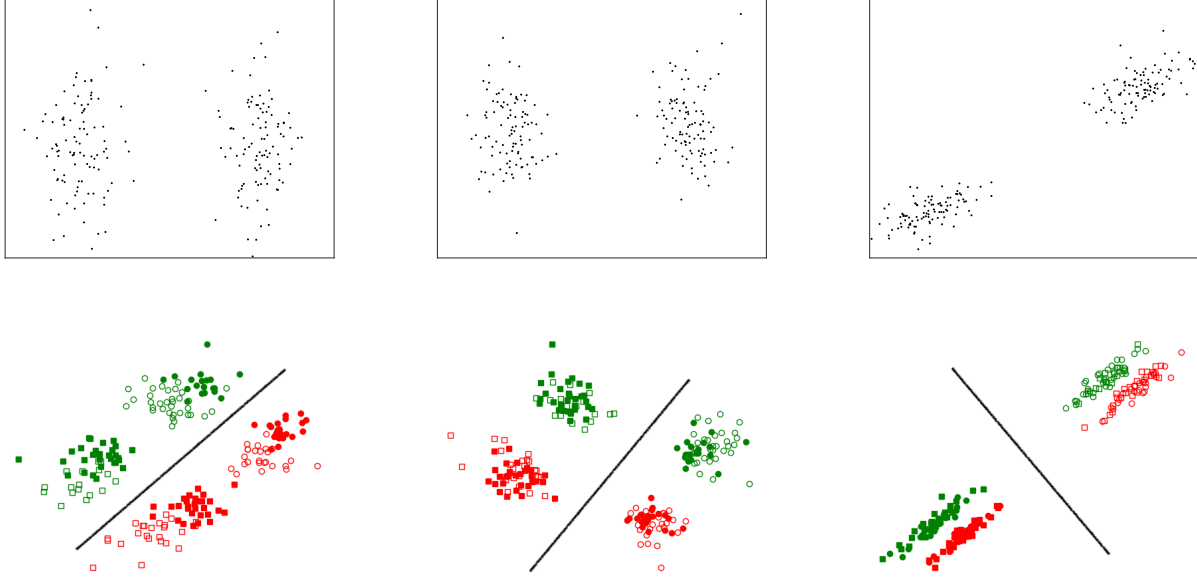


Fig. 6: This first row of this figure presents the original clusters, each subfigure contains the same number of points in 2D with 2 well distinguishable clusters. The second row presents the results of the application of Algorithm 2 for this dataset.

problems. That is, if one has a weighted/unweighted graph and wants to visualize it in 2D or 3D such that the vertices with smaller weights are put close to each other and vertices with larger weights are drawn far from each other, we can use MDS on the adjacency matrix. Note that if two vertices are not connected one can use all pairs shortest path between these vertices to define distance.

To demonstrate such an example we create two graphs with 100 vertices: one grid and one path; see the first row of Figure 7. The goal is to apply our proposed algorithm and see what would be the 3D visualization of such an input such that one projection demonstrates the first graph and the second one demonstrates the second graph.

The second row of Figure 7 demonstrates the application of Algorithm 1 with 2 fixed projections  $E1$  and  $E2$  from (4). Note that the the algorithm recovers both grid (the left subfigure) and the path (thr right subfigure).

The third row of Figure 7 demonstrates the results of the application of Algorithm 2 for the dataset described above. In this case again, the algorithm recovers the grid very well (the left subfigure) and the path (the right subfigure) is more clear than the one for fixed projections.

## 5 Conclusions and Future Work

We described a generalization of MDS which can be used to simultaneously optimize multiple distance functions defined on the same set of objects. The result is an embedding in 3D space with a set of given or computed projections that show the different views. This approach has applications for visualizing abstract data as well as multivariate networks. Our initial implementation relies on standard gradient descent, which is expensive given the additional overhead of simultaneous optimizations. We plan to implement a stochastic gradient descent version as well as consider multiview stress majorization.

## Acknowledgement

This research has been supported by National Science Foundation TRIPDS program, grant CCF-1740858.

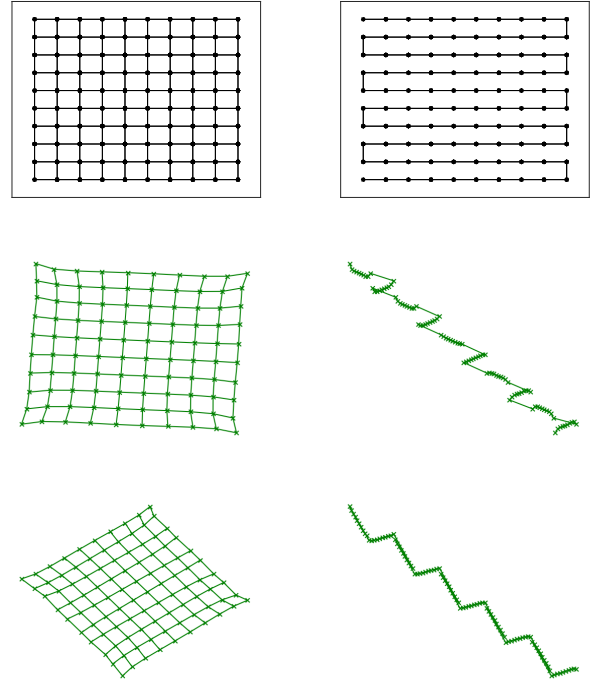


Fig. 7: The first row of this figure presents 2 graphs with 100 vertices, the first one is a path and the second one is a grid. The \* corresponds to vertices and edges connect between them. The second row demonstrates the application of Algorithm 1 to this dataset for projections  $E1$  and  $E2$ . The last row demonstrates the results of the application of Algorithm 2.

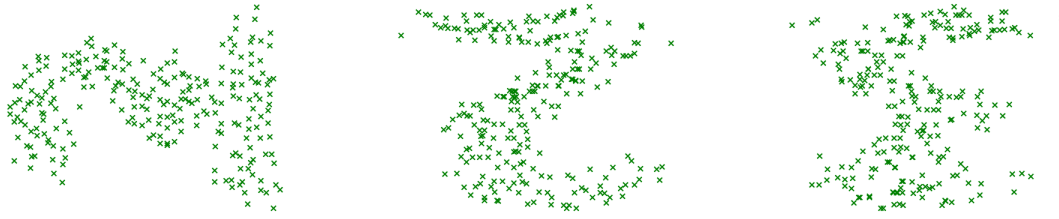


Fig. 8: This figure shows the results of the application of Algorithm 1 to the One-Two-Three dataset for projections from (5).

## References

- [1] P. Angelini, M. Geyer, M. Kaufmann, and D. Neuwirth. On a tree and a path with no geometric simultaneous embedding. In *International Symposium on Graph Drawing*, pp. 38–49. Springer, 2010.
- [2] D. Auber, D. Archambault, R. Bourqui, M. Delest, J. Dubois, A. Lambert, P. Mary, M. Mathiaut, G. Mélançon, B. Pinaud, et al. Tulip 5, 2017.
- [3] M. Bastian, S. Heymann, M. Jacomy, et al. Gephi: an open source software for exploring and manipulating networks. *ICWSM*, 8:361–362, 2009.
- [4] V. Batagelj and A. Mrvar. Pajek-program for large network analysis. *Connections*, 21(2):47–57, 1998.
- [5] A. Bezerianos, F. Chevalier, P. Dragicevic, N. Elmquist, and J.-D. Fekete. Graphdice: A system for exploring multivariate social networks. In *Computer Graphics Forum*, vol. 29, pp. 863–872. Wiley Online Library, 2010.
- [6] T. Bläsius, S. G. Kobourov, and I. Rutter. Simultaneous embedding of planar graphs. In R. Tamassia, ed., *Handbook of Graph Drawing and Visualization*, pp. 349–381. CRC Press, 2013.
- [7] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT’2010*, pp. 177–186. Springer, 2010.
- [8] 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*, 36(2):117–130, 2007.
- [9] L. Chen and A. Buja. Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. *Journal of the American Statistical Association*, 104(485):209–219, 2009.
- [10] I. F. Cruz and J. P. Twarog. 3d graph drawing with simulated annealing. In *International Symposium on Graph Drawing*, pp. 162–165. Springer, 1995.
- [11] E. Di Giacomo, W. Didimo, M. J. van Kreveld, G. Liotta, and B. Speckmann. Matched drawings of planar graphs. *J. Graph Algorithms Appl.*, 13(3):423–445, 2009.
- [12] J. Ellson, E. R. Gansner, E. Koutsofios, S. C. North, and G. Woodhull. Graphviz - open source graph drawing tools. In *Graph Drawing*, pp. 483–484, 2001.
- [13] N. Elmquist, P. Dragicevic, and J.-D. Fekete. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE transactions on Visualization and Computer Graphics*, 14(6):1539–1548, 2008.
- [14] E. R. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In *International Symposium on Graph Drawing*, pp. 239–250. Springer, 2004.
- [15] M. Geyer, M. Kaufmann, and I. Vrto. Two trees which are self-intersecting when drawn simultaneously. In *International Symposium on Graph Drawing*, pp. 201–210. Springer, 2005.
- [16] M. Ghoniem, F. Mcgee, G. Melançon, B. Otjacques, and B. Pinaud. The state of the art in multilayer network visualization. *arXiv preprint arXiv:1902.06815*, 2019.
- [17] L. Grilli, S.-H. Hong, G. Liotta, H. Meijer, and S. K. Wismath. Matched drawability of graph pairs and of graph triples. *Computational Geometry*, 43(6-7):611–634, 2010.
- [18] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inform. Process. Lett.*, 31:7–15, 1989.
- [19] A. Kerren, H. C. Purchase, and M. O. Ward. Introduction to multivariate network visualization. In *Multivariate Network Visualization*, pp. 1–9. Springer, 2014.
- [20] M. Kivelä, A. Arenas, M. Barthélemy, J. P. Gleeson, Y. Moreno, and M. A. Porter. Multilayer networks. *Journal of complex networks*, 2(3):203–271, 2014.
- [21] S. G. Kobourov. Force-directed drawing algorithms. In R. Tamassia, ed., *Handbook of Graph Drawing and Visualization*, pp. 383–408. CRC Press, 2013.
- [22] S. G. Kobourov and K. Wampler. Non-euclidean spring embedders. *IEEE Transactions on Visualization and Computer Graphics*, 11(6):757–767, 2005.
- [23] Y. Koren and L. Carmel. Visualization of labeled data using linear transformations. In *IEEE Symposium on Information Visualization 2003*, pp. 121–128. IEEE, 2003.
- [24] A. Koutsoudis, B. Vidmar, G. Ioannakis, F. Arnaoutoglou, G. Pavlidis, and C. Chamzas. Multi-image 3d reconstruction data evaluation. *Journal of Cultural Heritage*, 15(1):73–79, 2014.
- [25] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.
- [26] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11):2579–2605, 2008.
- [27] T. Munzner. Exploring large graphs in 3d hyperbolic space. *IEEE Computer Graphics and Applications*, 18(4):18–23, 1998.
- [28] C. Pich. *Applications of multidimensional scaling to graph drawing*. PhD thesis, 2009.
- [29] A. J. Pretorius and J. J. Van Wijk. Visual inspection of multivariate graphs. In *Computer Graphics Forum*, vol. 27, pp. 967–974. Wiley Online Library, 2008.
- [30] Y. Quéau, J. Mélou, J.-D. Durou, and D. Cremers. Dense multi-view 3d-reconstruction without dense correspondences. *arXiv preprint arXiv:1704.00337*, 2017.
- [31] R. N. Shepard. The analysis of proximities: multidimensional scaling with an unknown distance function. *Psychometrika*, 27(2):125–140, 1962.
- [32] B. Shneiderman and A. Aris. Network visualization by semantic substrates. *IEEE transactions on visualization and computer graphics*, 12(5):733–740, 2006.
- [33] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International journal of computer vision*, 80(2):189–210, 2008.
- [34] 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 research challenges. In *Computer graphics forum*, vol. 30, pp. 1719–1749. Wiley Online Library, 2011.
- [35] Y. Wang, Y. Wang, Y. Sun, L. Zhu, K. Lu, C.-W. Fu, M. Sedlmair, O. Deussen, and B. Chen. Revisiting stress majorization as a unified framework for interactive constrained graph visualization. *IEEE transactions on visualization and computer graphics*, 24(1):489–499, 2018.
- [36] M. Wattenberg. Visual exploration of multivariate graphs. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp. 811–819. ACM, 2006.

## A Multiview-MDS Gradients

In this appendix, we present and derive some of the formulas for the various gradient functions that are used in our 3DMGV algorithms. Our purpose is to assist the reader who wishes to implement these algorithms. Computation of the gradient of the multiview-MDS stress function (2) with respect to the positions  $X$  or orthogonal matrices  $Q_l$  is cumbersome if the right matrix calculus tools are not used. We begin with an overview of the matrix calculus tools that we need and then derive the gradient functions used in our implementations.

The results are derived for general embedding dimension  $p$ , projection rank  $r$ , and number of projections  $L$  (whereas we limited the exposition in Sections 2 and 3 to  $p = 3$ ,  $r = 2$  and  $L = 3$ ). The observed  $n \times n$  dissimilarity/distance matrices are denoted by  $D^1, D^2, \dots, D^L$ . The corresponding  $p \times p$  projection matrices  $\Pi_1, \Pi_2, \dots, \Pi_L$  can be written as  $\Pi_l = Q_l Q_l^T$ , where  $Q_l$  belongs to the set of  $p \times r$  orthonormal matrices  $\mathbb{O}^{p \times r}$ . The position matrix  $X \in \mathbb{R}^{n \times p}$  organizes the positions  $x_1, x_2, \dots, x_n \in \mathbb{R}^p$  by row.

### A.1 Some Matrix Calculus

We begin with an overview of some matrix calculus tools that we use in our derivations of the gradient functions of the multiview-MDS stress function (2).

Let  $y = f(X)$ , where  $f : \mathbb{R}^{n \times p} \rightarrow \mathbb{R}$ . The derivative of  $y$  with respect to  $X$  is the  $p \times n$  matrix given by

$$\frac{dy}{dX} := \left[ \frac{\partial f}{\partial X_{ji}} \right]_{ij},$$

where  $X_{ij}$  is the  $(i, j)$  entry of  $X$ . The gradient of  $y$  with respect to  $X$  is the  $n \times p$  matrix

$$\nabla_X y := \left[ \frac{\partial f}{\partial X_{ij}} \right]_{ij} = \left( \frac{dy}{dX} \right)^T.$$

We use derivative notation to derive the results that we need before changing to gradient notation.

The linearization of  $f$  about  $X_0$  is given by

$$\begin{aligned} y(X_0 + \Delta X) &= y(X_0) + \sum_{i,j} (\nabla_X y|_{X_0})_{ij} (\Delta X)_{ij} + \dots \\ &= y(X_0) + \text{tr} \left( \frac{dy}{dX} (X_0) \Delta X \right) + \dots \end{aligned}$$

(to save space, we do not explicitly write the term  $\mathcal{O}(\|\Delta X\|_F^2)$ , which in terms of differentials takes the form

$$dy = \text{tr} \left( \frac{dy}{dX} (X_0) dX \right).$$

We can use properties of differentials and the trace function to simplify the derivation of the various gradient functions.

If  $g : \mathbb{R} \rightarrow \mathbb{R}$  and  $z = g(f(X))$ , the chain rule says that

$$\frac{dz}{dX}(X) = g'(f(X)) \frac{df}{dX}(X).$$

If  $\Pi$  is a fixed  $p \times p$  matrix and  $y(X) = f(X\Pi^T)$ , we have

$$\begin{aligned} y(X + \Delta X) &= f((X + \Delta X)\Pi^T) \\ &= f(X\Pi^T + \Delta X\Pi^T) \\ &= f(X\Pi^T) + \text{tr} \left( \frac{df}{dX}(X\Pi^T) \Delta X\Pi^T \right) + \dots \\ &= f(X\Pi^T) + \text{tr} \left( \Pi^T \frac{df}{dX}(X_0\Pi^T) \Delta X \right) + \dots \end{aligned}$$

so that

$$\frac{d}{dX} (f(X\Pi^T)) = \Pi^T \frac{df}{dX}(X\Pi^T)$$

and

$$\nabla_X (f(X\Pi^T)) = \nabla f(X\Pi^T) \Pi.$$

Similarly, if we set  $\Pi = QQ^T$  and differentiate  $y(Q) = f(XQQ^T)$  with respect to  $Q$ , it can be shown that

$$\begin{aligned} \nabla_Q (f(XQQ^T)) &= \left( \left( \nabla f(XQQ^T) \right)^T X + X^T \nabla f(XQQ^T) \right) Q. \end{aligned}$$

We will use the two previous results in deriving formulas for the  $X$  and  $Q_l$  gradients of (6).

### A.2 Computation of Relevant Gradients

We begin by computing the  $X$  and  $Q_l$  gradients of the components  $\|\Pi_l x_i - \Pi_l x_j\|$  of the multiview-MDS stress function (6). The formulas for the  $X$  and  $Q_l$  gradients of (6) follow easily after that.

Let  $d_{ij}(X)$  be the distance between positions  $x_i$  and  $x_j$ . This can be written as

$$d_{ij}(X) = \|X^T e_i - X^T e_j\|_2 = \|X^T (e_i - e_j)\|_2,$$

where  $e_i, e_j$  are the  $i$ -th and  $j$ -th standard basis (column) vectors of  $\mathbb{R}^n$ . The square distance can be written as

$$\begin{aligned} d_{ij}^2(X) &= \|X^T (e_i - e_j)\|_2^2 \\ &= \text{tr} (X^T (e_i - e_j)(e_i - e_j)^T X) , \\ &= \text{tr} (X^T A_{ij} X) \end{aligned}$$

where

$$A_{ij} := (e_i - e_j)(e_i - e_j)^T.$$

Note that

$$\begin{aligned} \text{dtr} (X^T A_{ij} X) &= \text{tr} (d(X^T A_{ij} X)) \\ &= \text{tr} (dX^T A_{ij} X + X^T A_{ij} dX) \\ &= \text{tr} (X^T A_{ij}^T dX + X^T A_{ij} dX) , \\ &= \text{tr} ((2X^T A_{ij}) dX) \end{aligned}$$

and so

$$\frac{dd_{ij}^2}{dX}(X) = 2X^T A_{ij}.$$

It then follows that

$$\begin{aligned} \frac{dd_{ij}}{dX}(X) &= \frac{d}{dX} \sqrt{d_{ij}^2(X)} \\ &= \frac{1}{2\sqrt{d_{ij}^2(X)}} \frac{dd_{ij}^2}{dX}(X) , \\ &= \frac{1}{d_{ij}(X)} X^T A_{ij} \end{aligned}$$

and that

$$\nabla d_{ij}(X) = \frac{1}{d_{ij}(X)} A_{ij} X.$$

If  $\Pi$  is a  $p \times p$  matrix, we have

$$\begin{aligned} \nabla_X (d_{ij}(X\Pi^T)) &= \nabla_X d_{ij}(X\Pi^T) \Pi \\ &= \left( \frac{1}{d_{ij}(X\Pi^T)} A_{ij} (X\Pi^T) \right) \Pi . \\ &= \frac{1}{d_{ij}(X\Pi^T)} A_{ij} X \Pi^T \Pi \end{aligned}$$

If we set  $\Pi = QQ^T$  and differentiate with respect to  $Q$ , we similarly obtain

$$\begin{aligned} \nabla_Q (d_{ij}(XQQ^T)) &= \frac{1}{d_{ij}(XQQ^T)} (QQ^T X^T A_{ij} X + X^T A_{ij} XQQ^T) Q. \end{aligned}$$

For a fixed  $n \times n$  dissimilarity/distance matrix  $D$ , the MDS stress function can be written as

$$S(X; D) = \sum_{i < j} (d_{ij}(X) - D_{ij})^2.$$



which has the gradient

$$\begin{aligned}\nabla S(X; D) &= \nabla_X \sum_{i < j} (d_{ij}(X) - D_{ij})^2 \\ &= 2 \sum_{i < j} (d_{ij}(X) - D_{ij}) \nabla_X d_{ij}(X) , \\ &:= B(X; D)X\end{aligned}$$

where

$$B(X; D) := 2 \sum_{i < j} \frac{(d_{ij}(X) - D_{ij})}{d_{ij}(X)} A_{ij}.$$

If  $\Pi$  is a  $p \times p$  (orthogonal projection) matrix, then the matrix  $X\Pi^T$  is the  $n \times p$  matrix whose rows are equal to  $\Pi x_i$ . The gradient of the multiview MDS stress function with respect to  $X$  is

$$\begin{aligned}\nabla_X (S(X\Pi^T; D)) &= \nabla S(X\Pi^T; D)\Pi \\ &= B(X\Pi^T; D)X\Pi^T\Pi .\end{aligned}$$

If  $\{(\Pi_k, d_k)\}_{k=1}^K$  are  $k$  pairs of  $p \times p$  transformations and  $n \times n$  distance matrices, then the multiview-MDS stress function is

$$\begin{aligned}S_M(X; \{(\Pi_l, D_l)\}_{l=1}^L) &:= \sum_{l=1}^L S(X\Pi_l^T; D_l) \\ &= \sum_l \sum_{i < j} (d_{ij}(X\Pi_l^T) - (D_l)_{ij})^2\end{aligned}$$

and its gradient is

$$\begin{aligned}\nabla_X S_M(X; \{(\Pi_l, D_l)\}_{l=1}^L) &= \sum_l \nabla_X S(X\Pi_l^T; D_l) \\ &= \sum_l B(X\Pi_l^T; D_l)X\Pi_l^T\Pi_l .\end{aligned}$$

This is the expression for the gradient function that we use in Algorithm 1 and as part of Algorithm 2.

The gradient of the MDS stress function with respect to  $Q$  can be computed similarly,

$$\begin{aligned}\nabla_Q S(XQQ^T; D) &= \\ \left( QQ^T X^T \left( B(XQQ^T; D) \right)^T X + X^T B(XQQ^T; D) XQQ^T \right) Q.\end{aligned}$$

This is the expression for the gradient function that we use in Algorithm 3.

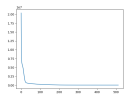


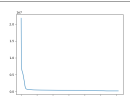
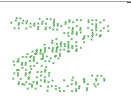
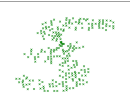
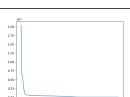

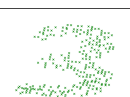
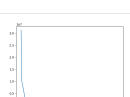

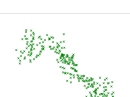

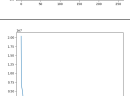

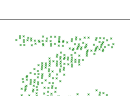
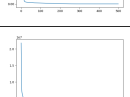


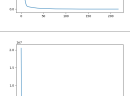


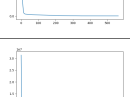

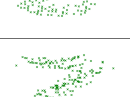
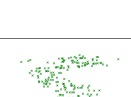
123-12p1				
123-23p1				
123-13p1				
123-123p1				
123-12p2				
123-23p2				
123-13p2				
123-123p2				

Table 1: Fixed Projections

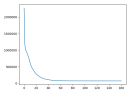
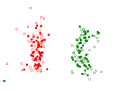
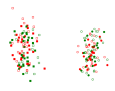
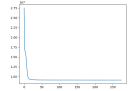
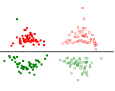
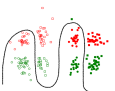
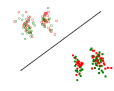
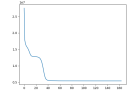

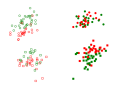
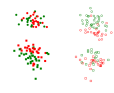
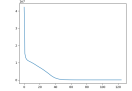
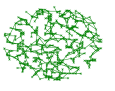
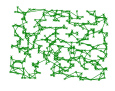
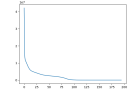
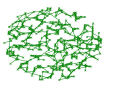
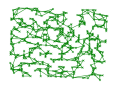
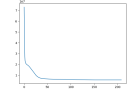
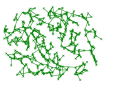
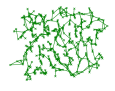
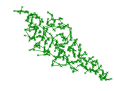
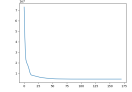
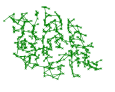
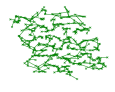

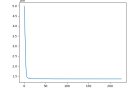
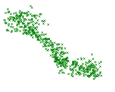


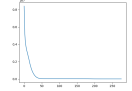
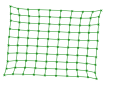
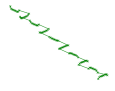
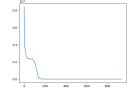
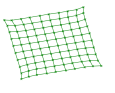

cluster-12p1				
cluster-123p1				
cluster-123p2				
sqcirtr-12p1				
sqcirtr-12p2				
sqcirtr-123p1				
sqcirtr-123p2				
123old-123p1				
grid-12p1				
grid-12p2				

Table 2: Fixed Projections

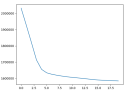



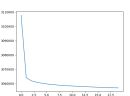
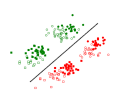
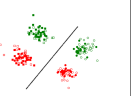
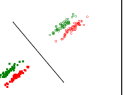
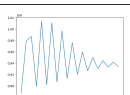
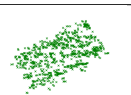
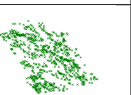
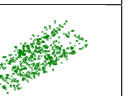
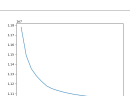
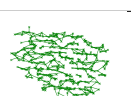
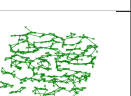
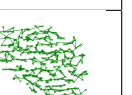
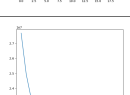
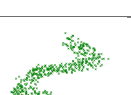

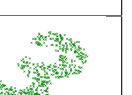
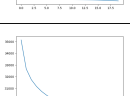
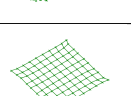

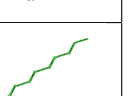
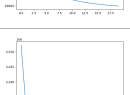


proj123-123p2				
projcluster-123p1				
projconfname-123p1				
projscirtr-123p2				
proj123old-123p1				
projgrid-12p1				
projcupmouse-12p1				

Table 3: variable projection