EG UK Computer Graphics & Visual Computing (2015)
Rita Borgo, Cagatay Turkay (Editors)

Explaining Neighborhood Preservation for Multidimensional Projections

Rafael Messias Martins[1,2] , Rosane Minghim[1] and A. C. Telea[2]

[1]Institute of Mathematics and Computer Science (ICMC), University of São Paulo, São Carlos, Brazil
[2]Johann Bernoulli Institute, University of Groningen, The Netherlands

Abstract

Dimensionality reduction techniques are the tools of choice for exploring high-dimensional datasets by means of low-dimensional projections. However, even state-of-the-art projection methods fail, up to various degrees, in perfectly preserving the structure of the data, expressed in terms of inter-point distances and point neighborhoods. To support better interpretation of a projection, we propose several metrics for quantifying errors related to neighborhood preservation. Next, we propose a number of visualizations that allow users to explore and explain the quality of neighborhood preservation at different scales, captured by the aforementioned error metrics. We demonstrate our exploratory views on three real-world datasets and two state-of-the-art multidimensional projection techniques.

Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computer Graphics]: Picture/Image Generation—Viewing algorithms, I.4.10 [Computer Graphics]: Image Representation—Multidimensional

1. Introduction

Dimensionality reduction (DR) techniques transform, or project, high-dimensional datasets into low-dimensional ones which can be more easily analyzed, interpreted, and visualized. While projections cannot preserve (and show) all original high-dimensional information, they keep much of the so-called *data structure* in the low-dimensional space. This allows one next to search the projection for structures such as clusters, outliers, correlations, and trends, by using various visualization techniques. Projections are heavily used in many data mining and exploration tasks in many application domains such as medical science, business intelligence, and security.

Besides information loss in the sense of not showing all original data dimensions, projections are also challenged by precision loss, in the sense of errors that affect the projected data structure. Two main types of such errors exist: *Distance errors* make distances between projected points inaccurately reflect distances between the original high-dimensional points. *Neighborhood errors* make the nearest-neighbors of a point be different in the original and projected space. Both types of errors in turn can significantly affect the insights obtained on the data at hand when using the projection as a proxy to reason about the high-dimensional visual space in terms of finding and comparing clusters, outliers, and trends [VDHM97, Aup07, SvLB10, LA11, vdMH12].

We address the task of interpreting projections by making explicit where neighborhood-related errors appear. For this, we propose several metrics to quantify the appearance of such errors in projections and introduce visualizations that allow selecting suitable scales or levels-of-detail to examine such errors, and support users in understanding and using the projection in their presence. Our explanatory methods are simple to implement, computationally scalable and can be easily integrated in classical scatterplot views of any projection technique, including linear [Tor65, BTB13] and non-linear ones [PNML08, vdMH08, TMN03, PSN10], in a *black box* fashion. That is, we only need to access the input high-dimensional points and the output low-dimensional projections thereof, and need no details of, or access to, the projection internals. This makes adding our techniques easy to any projection-based application.

2. Related Work

For a dataset $D^n = \{\mathbf{p}_i \in \mathbb{R}^n\}_{1 \le i \le N}$ of N n-dimensional points, a DR technique can be seen as a function $f : \mathbb{R}^n \to \mathbb{R}^m$ which maps each $\mathbf{p}_i \in D^n$ to a point $\mathbf{q}_i \in D^m$, so as to keep the so-called *data structure* as similar as possible in $\mathbb{R}^n$ and $\mathbb{R}^m$. Here, n is typically large (tens up to thousands of dimensions), and m is typically 2 or, more rarely, 3.

Projection techniques: Tens of different DR techniques exist, able to different extents to preserve various data-structure aspects. For instance, Multidimensional Scaling (MDS) methods compute f by optimizing for a low normalized stress $\sigma = \sum_{i,j}(\|\mathbf{p}_i - \mathbf{p}_j\| - \|\mathbf{q}_i - \mathbf{q}_j\|)^2 / \sum_{i,j}(\|\mathbf{p}_i - \mathbf{p}_j\|)^2$ [BG05]. This avoids having to access the full D^n coordinate data. However, computing and storing distances is $O(N^2)$ for N points.

Landmarks MDS [dST04] and Pivot MDS [BP07] speed-up projection by using MDS on a subset of representative points and projecting the remaining points by local interpolation. Fastmap [FL95] achieves $O(N)$ linear complexity but has a worse stress minimization. Nonlinear optimization methods iteratively search the projection parameter space to minimize the stress σ [GKN04, Sam69] using, for instance, force-based relaxation [TMN03], similar to graph layout methods [Ead84]. To speed up computations, LSP projects only a subset of control points from D^n and fits the remaining ones by Laplacian smoothing [PNML08]. To increase projection quality, LAMP extends the above idea to allow users to interactively place control points [JPC*11].

Projection errors: Given the huge range of projection techniques, it is hard for users to assess the nature, magnitude, and location of projection errors they introduce, without supporting tools. Understanding errors is crucial to correctly interpreting high-dimensional data by means of a projection [vdMH12, MCMT14, SvLB10, Aup07], and is done at three levels of detail. At the coarsest level, error metrics like normalized stress [BG05], correlation coefficients [GZZ05], and silhouette coefficients [PNSK*06] capture the overall projection quality by a single number. This allows globally comparing projection methods [PPM*15, EMdS*15], but does not show how errors are spread over the various points in a given projection. On the next detail level, distance scatterplots show the correlation of distances in D^n *vs* D^m for every point-pair [JPC*11] and neighborhood preservation plots show how a projection preserves neighborhoods, for all possible neighborhood sizes [PNML08, VDHM97, BP92, VK01]. Of these, the trustworthiness-preservation metric [VK01] stands out as it shows not only how many neighbors are preserved when projecting, but also if the order of neighbors changes. However, just as distance scatterplots, these give an aggregated insight, and do not show how projection errors are spread *per projected point* $\mathbf{q}_i$.

At the finest detail level, several methods show where in the projection D^m distance- and/or neighborhood-preservation errors occur, and how large these are. Van der Maaten *et al.* use the t-SNE technique [vdMH08] to create a set of 2D projections, each showing a different distance-errors per projected point $\mathbf{q}_i$ [vdMH12]. Aupetit *et al.* show, for a given $\mathbf{q}_i$, distance stretching and compression metrics, telling if $\mathbf{q}_i$ was projected too close or too far from all other $\mathbf{q}_{j\neq i}$. However, this requires one to select a point of interest $\mathbf{q}_i$ to explore. Martins *et al.* show false and missing neighbors, defined in terms of points projected too close, respectively too far, from their neighbors [MCMT14]. However, this chiefly measures distance preservation, and not neighborhood preservation.

In contrast to distance-preservation metrics, *neighborhood-preservation* metrics aim to find how k-nearest neighborhoods are affected by the projection, and ideally must cover two cases [VK01]: preservation of D^n neighborhoods (are neighbors in D^n projected to neighbors in D^m?) and trustworthiness of D^m neighborhoods (are neighbors in D^m also neighbors in D^n?). For this, Schreck *et al.* compute, for each $\mathbf{p} \in D^n$, a projection precision score (*pps*) defined as the normalized distance between two k-dimensional vectors containing the distances between $\mathbf{p}_i$ and its k nearest neighbors in D^n, respectively $\mathbf{q}_i$ and its k nearest neighbors in D^m [SvLB10]. Color mapping the *pps* atop the projection shows areas with poorly preserved neighborhoods. This covers the first neighborhood question outlined above but not the second one, leaving important errors, *e.g.* false neighbors in terms of [MCMT14], undetected. Motta *et al.* propose a neighborhood validation metric, which combines precision (how many neighbors in D^m are also neighbors in D^n) and recall (how many D^n neighbors are also neighbors in D^m) into an error called the *F-measure* [MMLO15]. While related to our set-difference view (Sec. 3.2), F-measures are computed by using an extended MST graph [MLN*13] to define neighborhoods in D^n and D^m, while we use the simpler to compute, and more intuitive, k-nearest neighbors.

Neighborhood-preservation metrics have a salient advantage over distance-error metrics: Many data analyses using projections focus on finding point groups (clusters) and outliers. For such tasks, actual *distances* between points are less important than *neighbors*. Indeed, we visually decide that a point-set forms a cluster by typically using the fact that inter-point distances over the cluster are much smaller than distances between the cluster and other points. Hence, understanding neighborhood-preservation errors is at least as important as understanding absolute-distance errors. In this context, our main contributions are (1) three neighborhood-preservation metrics that adapt [MCMT14, VK01] to find and interpret false and missing neighbors for different neighborhood sizes given by k-nearest neighbors; (2) three corresponding multiscale views that allow exploring neighborhood-preservation errors at the desired (local) level of detail, based on the projection's visual topology. These are presented next.

3. Visual Exploration Method

We define the k-neighborhood of a point $\mathbf{x}$ as the list $\nu_k(\mathbf{x}) \subset \{1,\ldots,N\}$ of IDs of the k-nearest neighbors of $\mathbf{x}$ (for a given k), sorted increasingly on Euclidean distance. When projecting a high-dimensional dataset D^n into m dimensions, each point i has two neighborhoods – one in D^n, denoted by $\nu_k^n(i)$, and the other in D^2, denoted by $\nu_k^2(i)$. All our examples next use 2D projections ($m = 2$) and Euclidean distances, for illustration simplicity. However, all our techniques can be equally easily applied to 3D projections and/or other distance metrics.

To analyze neighborhood preservation for a point i, we identify three important neighbor sets for this point:

- **missing neighbors** $= \{\mathbf{p}_j \in \nu_k^n(i) \wedge \mathbf{q}_j \notin \nu_k^2(i)\}$
  These are points that, while present in $\nu_k^n(i)$, were found not very important, and thus pushed outside $\nu_k^2(i)$. Missing neighbors are, thus, not found when visually examining the projection around point i;
- **false neighbors** $= \{\mathbf{p}_j \notin \nu_k^n(i) \wedge \mathbf{q}_j \in \nu_k^2(i)\}$
  These points were originally far away from point i (outside $\nu_k^n(i)$), but were brought close to i in the projection. False neighbors are, thus, points we visually see as being close to point i in the projection, but which are in reality far from point i in the high-dimensional space;
- **true neighbors** $= \{\mathbf{p}_j \in \nu_k^n(i) \wedge \mathbf{q}_j \in \nu_k^2(i)\}$
  These are points which we visually find close to point i in

the projection, and are also close to the same point in high-dimensional space.

Recall that projections are used to reason about point neighborhoods in D^n by using point neighborhoods in D^2. As such, ideally, we do not want a projection to create any false or missing neighbors, as these would mislead users when interpreting the data. Ideally, a projection should only create true neighbors, *i.e.* $\nu_k^n(i) = \nu_k^2(i)$, for all points i and neighborhood sizes k. However, as we will see, even state-of-the-art projection techniques are far from this ideal.

To explore how much, where, and why projections deviate from ideal neighborhood-preservation, we next propose several views to analyze different neighborhood-preservation aspects. As a running example, we use the well-known *segmentation* dataset (2100 points, 18 dimensions) from the UCI Machine Learning Repository [Lic13]. Each point represents a small pixel-block extracted from 7 hand-segmented outdoor images. Dimensions encode various image descriptors, such as color and contrast histograms and edge detectors. This dataset is frequently used in infovis papers to assess the quality of projection techniques in terms of being able to cluster similar image structures [JPC*11, PNML08, MCMT14]. For projection, we use the well-known high-quality nonlinear LAMP technique [JPC*11, MCMT14].

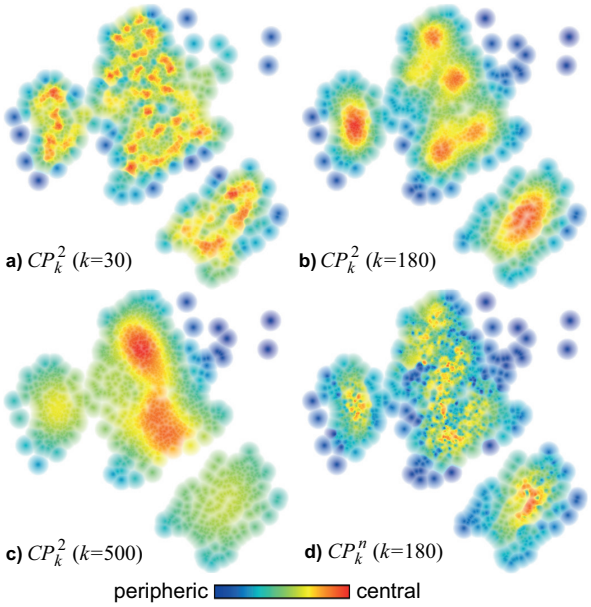


Figure 1: *Centrality preservation view,* segmentation *dataset, LAMP projection. (a-c) Centrality CP_k^2, for three neighborhood sizes k. (d) Centrality CP_k^n, for $k = 180$ neighbors.*

3.1. Centrality Preservation View

Given a set D of N points, we first introduce the *centrality preservation* metric

$$CP_k(j) = \sum_{1 \le i \le N, j \in \nu_k(i)} k - \rho_i(j), \qquad (1)$$

where $\rho_i(j)$ is the rank of point j in $\nu_k(i)$ when ordered from nearest ($\rho_i = 0$) to farthest ($\rho_i = k-1$). Points j that are close neighbors to many other points i of D get high $CP_k(j)$ values, such as points which are 'central' with respect to the structure of the set D; points close to the 'periphery' of D are not close neighbors to many other points, so they get low $CP_k(j)$ values. We visualize CP_k by color-coding its values over the 2D projection point-cloud using Shepard interpolation to fill in gaps between close points and thereby generate a continuous, easier to visually follow, color image. Full details of this technique are given in [MCMT14].

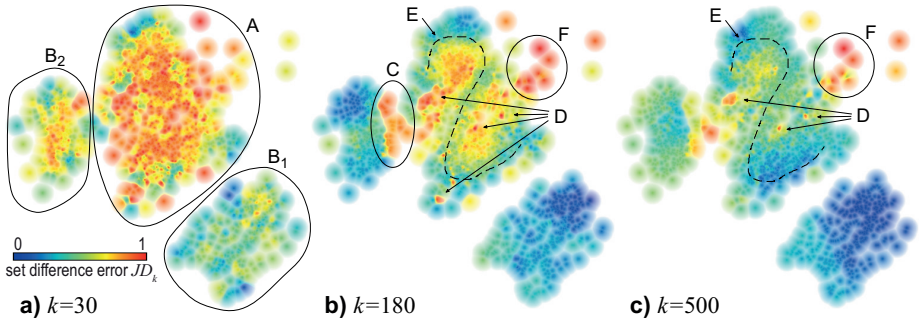
Consider now how CP_k^2, *i.e.* the centrality preservation computed over the 2D projection space, evolves (Figs. 1a-c): We see how central points have high values (red), while peripheral points have low values (blue). Visualizing CP_k^2 helps finding a good scale at which we want to assess neighborhood preservation next. As Fig. 1a shows, too low k values highlight very small changes in local point density. Assessing neighborhood preservation at such scales is, arguably, not interesting for most real-world scenarios – indeed, an even tiny shift in the points' positions would create slightly different CP_k^2 values. Conversely, setting too large k values highlights too coarse-scale patterns that do not match the shape of the projection (Fig. 1c). In-between values, *e.g.* $k = 180$ (Fig. 1b), highlight centrality patterns which match well the *shape* of the projection — we see how red bumps nicely match the main point-groups visible in the projection. Hence, we use the centrality view (showing CP_k^2) to select a k, or scale, which best matches the desired level-of-detail to explore the projection next, based on the match between the shapes we see in the projection and the CP_k^2 peaks.

Once we have a good k value, we next visualize CP_k^n *drawn over the projected points* D^2. To explain this design, consider a projection that perfectly preserves neighbors: In such a case, $CP_k^n(i) = CP_k^2(i), \forall i \in \{1,\ldots,N\}, \forall k \in \{1,\ldots,N\}$. Hence, the visualization of CP_k^n should match the already-understood pattern shown earlier by CP_k^2. Conversely, in areas where neighborhoods are not preserved, CP_k^n will show perturbations to this pattern: If points which were peripheral in D^n become central in D^2, then we will see blue points in central areas in D^2, where we expect red points; and if points which were central in D^n become peripheral in D^2, then we will see red points on the periphery of D^2, where we expect blue points. Fig. 1d shows this: Compared to Fig. 1b (both images are for $k = 180$), we see how Fig. 1d shows a somewhat similar trend of red central points and blue peripheral points. However, the smooth red-to-blue transition in Fig. 1b is partially lost in most areas of the image. This tells that there are many neighborhood-preservation errors *and* that these occur over most of the projection. Once signalled, such errors can be analyzed in more detail by the views presented next.

3.2. Set Difference View

The second view we propose, called the set-difference view, compares the D^n and D^2 neighbor-sets of each data point i using the Jaccard set-distance [LW71], by computing

$$JD_k(i) = 1 - \frac{\nu_k^2(i) \cap \nu_k^n(i)}{\nu_k^2(i) \cup \nu_k^n(i)}. \qquad (2)$$



a) k=30 **b)** k=180 **c)** k=500

Figure 2: *Set difference view,* segmentation *dataset, LAMP projection, using the same scales (k values) as in Fig. 1.*

This value represents the neighborhood preservation *error* of each point i. Its interpretation is simple: $JD_k(i) = 1$ tells that the D^n neighborhood of point i was completely lost by the projection, while $JD_k(i) = 0$ tells that the projection preserved the k neighbors of i perfectly. However, the neighbors' ranks, positions, and distances relative to point i are not considered. Fig. 2 shows three set-difference views for the same k values as in Fig. 1, for ease of comparison.

Since the Jaccard distance varies between 0 and 1, its values are easy to interpret: High JD_k values (warm colors) show poor neighborhood preservation; low values (cold colors) show areas where neighbors are well preserved. In addition to the centrality preservation view (Fig. 1), we get now extra insights, as follows. First, for the fine-grained scale $k = 30$ (Fig. 2a), we see a relatively low neighborhood preservation (high JD_k values) for most points, except the ones in the low-right group B_1. For our reference scale $k = 180$ (Fig. 2b), which is a good level-of-detail to examine this dataset, as explained in Sec. 3.1, we see that both smaller point-groups B_1 and B_2 have very good neighborhood preservation (JD_k low values). This confirms that the LAMP method was right to separate them from the central group A. At this scale we also see an 'isthmus' (Fig. 2b, marker C) connecting group B_2 with the large group A, with very high neighborhood-preservation errors. Interestingly, once we look left past this isthmus, group B_2 shows a very good neighborhood preservation (dark blue colors). This indicates that groups B_2 and A may, actually, not be *close* in the high-dimensional space, or, in other words, that we are looking at a projection artifact here. We will explore this hypothesis next with our other views (Sec. 3.3).

We also see several red islands in the central group A (Fig. 2b, zones D). Increasing k to 500, these islands are reduced to a few outliers (Fig. 2c, zones D). This tells that, on a coarse scale, group A has no neighbor-preservation issues, so it is *indeed* a group in D^n space. However, the isolated red outliers F remain red even at a coarse scale. This tells that these points are wrongly projected in the respective place, close to group A's right border. Finally, a more subtle observation can be done. Looking at group A in the projection, without the insight shown by the metric in the set-difference view, we may think of it as a compact large cluster of quite similar points. Yet, in the set-difference view, we see a Z-shaped 'corridor' of points of medium error (Fig. 2b, marker E), that winds through the high-error red islands. This suggests us that group A may not be that homogeneous in D^n space. We will explore this finding next using our subsequent views.

3.3. Sequence Difference View

While the set-difference view (Sec. 3.2) shows an easy-to-interpret picture of how many true neighbors a projected point has, it does not account for the *position* of these neighbors, or how they may have been reordered by the projection. To capture this, we propose a new *sequence difference* metric to compare the D^n and D^2 neighborhoods of a point i as

$$SD_k(i) = \frac{1}{2} \sum_{j \in \nu_k^2(i)} (k - \rho_i^2(j)) \cdot |\rho_i^2(j) - \rho_i^n(j)| + \frac{1}{2} \sum_{j \in \nu_k^n(i)} (k - \rho_i^n(j)) \cdot |\rho_i^2(j) - \rho_i^n(j)|, \quad (3)$$

where $\rho_i^d(j)$ is the rank of point j in the distance-sorted neighborhood $\nu_k^d(i)$ for $d \in \{2, n\}$. The metric's second term in each sum penalizes neighbors j which do not keep ranks after projection, *i.e.* $\rho_i^2(j) \neq \rho_i^n(j)$. The first term in each sum gives a higher weight to close neighbors. This captures the fact that not preserving the rank of a close neighbor is worse, in terms of interpretation of the resulting projection, than not preserving the rank of a very distant neighbor. This, in turn, models the way users typically interpret a projection, *i.e.* by locally scanning and querying small point neighborhoods to find what is most similar to a given point. Fig. 3 shows the sequence difference view, with four k scales, for our running example.

The four scales show much less differences as compared to the earlier error views (Figs. 1 and 2). This is expected, as SD_k (Eqn. 3) can be seen as a rank-weighted version of JD_k (Eqn. 2). To explain this, consider a neighborhood $\nu_k^n(i)$ and its 2D counterpart $\nu_k^2(i)$ which are identical, except that the farthest two neighbors are swapped. The resulting value $SD_k(i)$ will be equal to 2. If we take a slightly smaller neighborhood of $k-2$ elements, $SD_{k-2}(i)$ equals zero, since the first $k-2$ elements in both $\nu_k^n(i)$ and $\nu_k^2(i)$ are identical. Now, consider that $\nu_k^n(i)$ differs from $\nu_k^2(i)$ in terms of the first two elements being swapped. The resulting value $SD_k(i)$ will equal $2k$, a value much larger than 2. Hence, small changes at the border of neighborhoods, where new points are considered as k increases, have small impacts, which yields a smooth variation of SD_k as function of k.

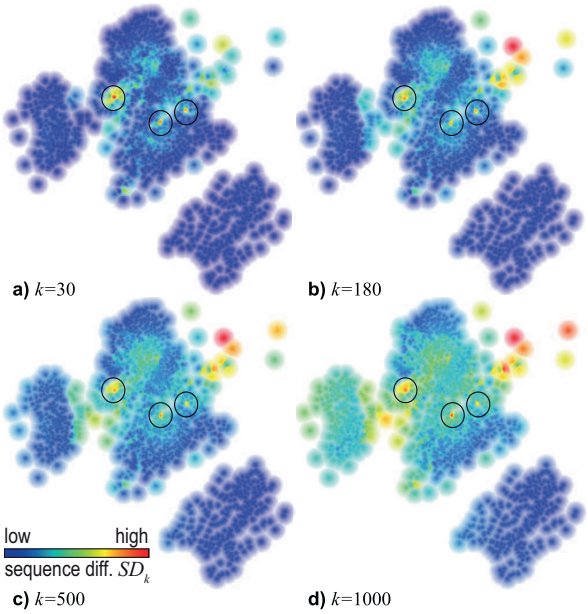


Figure 3: *Sequence difference view,* segmentation *dataset, LAMP projection, for four increasing scales (k values).*

The sequence-difference view (Fig. 3) highlights all high-error areas exposed by the set-difference view (Fig. 2b). However, as visible, the results are now much more stable for different k values. Separately, high-error outliers (small red dots marked in Fig. 3) are now much better visible than with the set-difference view, and at all scales.

3.4. Refining the Exploration

Using our set and sequence difference views, we discovered a salient border that separates the highly-coherent group of points in the left in our projection from the large central group via an isthmus of in-between points (see Figs. 2b and 3). We hypothesized that the left point-group was actually well separated in high-dimensional space from the central group, and that the linking isthmus was just an artifact of the projection. To verify this hypothesis, we will locally analyze the points close to this separation border. For this, we first select a point **q** to the right of the border, and color the *true neighbors* of **q** (see beginning of Sec. 3) by their ρ_q^n ranks, using a colormap which assigns red to close true neighbors and blue to far true neighbors in D^n; points which are not true neighbors are colored dark blue. We use here a brighter colormap than the one present in the set-difference and sequence-difference views, as we will also draw additional elements atop the color-coded points, as explained next. Fig. 4a shows the result.

Warm colors spread *only* to the right of the border, telling us that the nearest true neighbors of **q** are all placed to the right of the border, *i.e.*, points on the isthmus are far away, in D^n, from points in the left group. Additionally, we draw black edges between **q** and all its *missing neighbors*, map the ranks ρ_q^n to edge opacities, and use an edge bundling technique [HET12] to group close edges together, thereby reducing clutter. A similar design was used in [MCMT14] to visualize missing neighbors for distance-based errors. We see

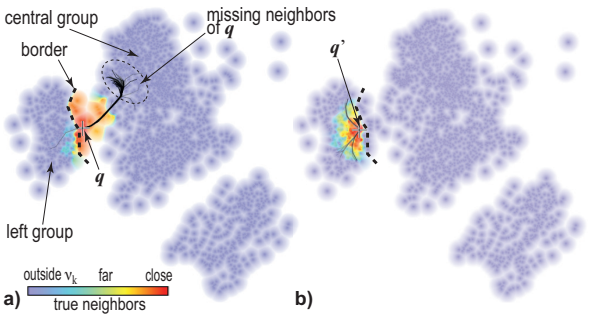


Figure 4: *Local analysis of the connection between the left and central groups. The visual border, seen also in Figs. 2b and 3, is marked by a dotted line.*

how the edge bundle emerging from the selected point **q** extends *only to the right* of the border (Fig. 4a). This tells that all neighbors of point **q** belong to the central cluster. Repeating the same operation for a point $\mathbf{q}'$ located on the *left* of the border shows us that all true and missing neighbors of $\mathbf{q}'$ are located in the left cluster. Together, these findings tell us that the border we discovered by our views is, indeed, the true border between the left and central clusters, an insight that might not be clear for the user of the projection without the investigation supported by the introduced tools.

Comparison: To better understand the added-value of our metrics, we show in Fig. 5 the *projection precision score (pps)* of Schreck *et al.* [SvLB10] for the same k values. For the first two k values (30,180), we hardly see any quality difference between different projection regions. For the last two k values (500,1000), some of the insights we saw in our proposed views appear a bit better, such as the isthmus between the left and central groups, a few high-error outliers, and the overall lower error of the bottom-right group. However, these views are much more sensitive to the chosen scale (k value) than ours, and the salience and separation of interesting patterns from noise is harder. Also, patterns such as the Z-shaped zone of low errors separated by high-error islands in the central point-group (Fig. 2) are not visible. Finally, since we have no tools to refine the exploration, we cannot check what the global insights presented by the *pps* view actually mean.

To see whether the found patterns match data-related insight, we show the projection colored by class – a categorical user-assigned attribute that maps every point in the *segmentation* dataset (an image block) to one of the 7 images it comes from (Fig. 6). We see how the class data, an attribute which was *not* used in the projection, shows very similar patterns to the ones found by our views: The right-bottom group has points in the same class (orange); the isthmus (cyan) linking the left group (dark blue) with the central group has a class border precisely where our views indicated neighborhood-preservation issues; and the Z-pattern is apparent if we follow the red-and-light-blue points mixed in the central group.

4. Additional Examples

To better outline the way of working of our projection inspection tools, we show next two exploration examples for two different datasets visualized by two different projections.

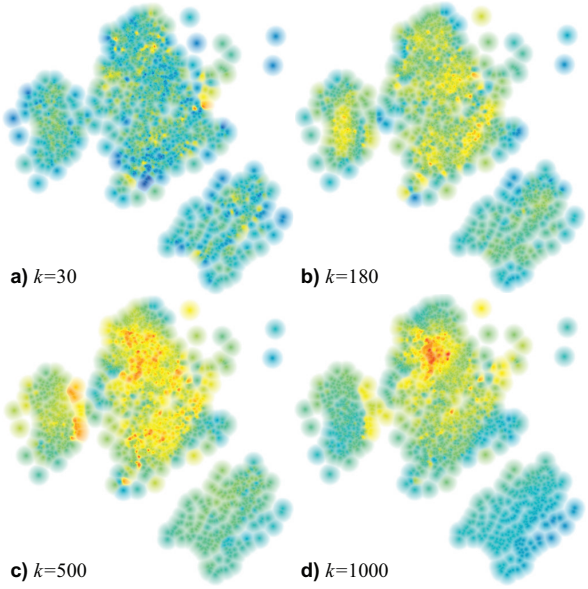


Figure 5: *Projection precision score (pps) [SvLB10], segmentation dataset, LAMP projection, for four scales (k values).*

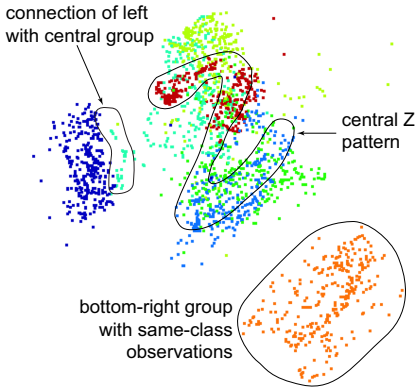


Figure 6: *Original class data for the* segmentation *dataset.*

Github dataset: This dataset has 725 observations, each describing one of the 1000 highest-ranked open source software projects from GitHub [Git15]. For each project, 30 software quality metrics are extracted, such as size (lines of code, file count), average coupling and cohesion of modules, complexity, and number of forks and open issues [LM06]. We visualize this 30-dimensional dataset using the LSP projection [PNML08], aiming to find separated clusters of projects with similar quality attributes. The projection shows a large central cluster, surrounded by a few outliers (Fig. 7). Our question is: Are projects indeed grouped into one similar set, or is the lack of separated clusters an artifact of the LSP technique?

To answer this question, we first select a suitable neighborhood size (k value), as explained in Sec. 3.1, and get a value $k = 72$. Next, we inspect the set-difference view (Fig. 7a). We see a quite poor neighborhood preservation for most points, and also find a group of points (G_1) with a large error; and a group of points (G_2) with a low error. The

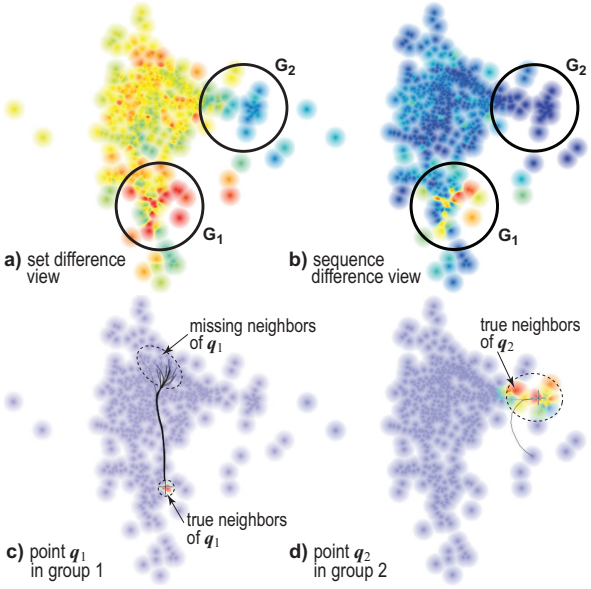


Figure 7: *Github dataset, LSP projection, $k = 72$ neighbors.*

sequence-difference view, next, shows that G_1 has the largest neighborhood-preservation error in the projection, while G_2 has a very low error (Fig. 7b). We next examine these groups in detail, as they seem to be the most important outliers which do not fit in the central cluster. For this, we select a point $\mathbf{q}_i$ in each group and color the projection by $\mathbf{q}_i$'s true neighbors and show $\mathbf{q}_i$'s missing neighbors with bundled edges (see Sec. 3.4). We can now explain the two outlier groups: In G_1, not only most of the missing neighbors are far away from the selected point, but all bundle edges are high-valued (dark), telling that these neighbors were very close to $\mathbf{p}_1$ in D^n (Fig. 7c). Also, the warm color spot around $\mathbf{q}_1$ is very small, telling that many of the neighbors of $\mathbf{q}_1$ in the projection are not true neighbors. This tells that the true G_1 has been 'scattered' over a large area by the projection. In contrast, when selecting a point $\mathbf{q}_2$ in G_2, we see almost no edges reaching out; also, most points around $\mathbf{q}_2$ are warm-colored, telling that most points in the projection around $\mathbf{q}_2$ are indeed its true neighbors (Fig. 7d). Hence, G_2 is indeed a 'true' group in D^n. Since G_2 is relatively well separated from the central group, we conclude that it represents a few highly-similar *and* quite different software projects from the rest of the analyzed set.

Corel dataset: This dataset has 1000 points, each representing a photograph described by 150 SIFT features, common in image analysis [LW03]. We construct a projection of the data using LAMP and obtain the star shape image in Fig. 8. This suggests that there are several quite well separated image classes, one for each star branch, and a clump of 'average' images (star's central region). To verify that the projection is indeed correct (and thus our interpretation of the data is correct), we use our views. First, we use the centrality view to select a suitable neighborhood size for the analysis, which yields a value $k = 200$. Next, we inspect the set-difference and sequence-difference views (Fig. 8a,b). The set-difference

view shows a quite high neighborhood-preservation error, spread uniformly over the projection (Fig. 8a). This is a first sign that the projection may have problems. The sequence-difference view allows us to find out high-error and low-error projection areas (Fig. 8b). This shows us that distance units (in 2D space) *along* two different star branches do not map the same distance from D^n space. Note that, without this insight, the star-shaped projection would be arguably interpreted differently – in particular, it would be hard for the user to know that some branches represent very similar images, while others capture far less similar images.

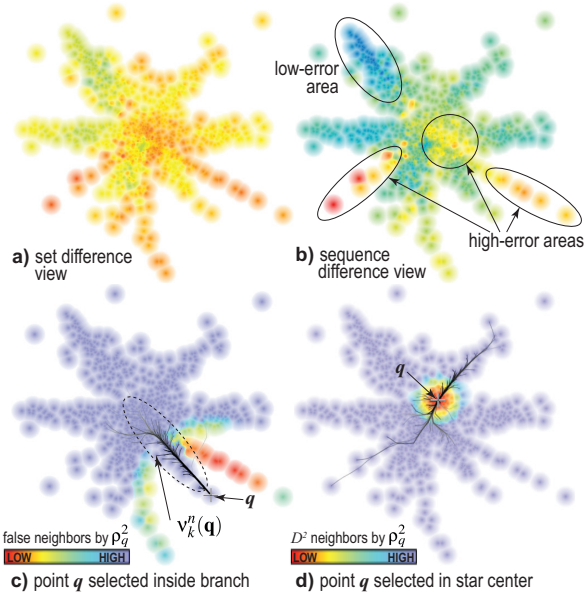


Figure 8: *Corel dataset, LAMP projection, $k = 200$ neighbors.*

This finding makes us ask next whether D^2 neighbors *between* star branches represent the D^n neighborhood as well as D^2 neighbors *along* star branches. To check this, we select a point $\mathbf{q}$ on a star branch having low neighborhood-preservation error, and color-code the *false neighbors* of $\mathbf{q}$ (see the beginning of Sec. 3) by their rank ρ_q^2 (nearest to farthest, as defined in Sec. 3.1). Additionally, we show the whole D^n neighbor-set $\nu_k^n(\mathbf{q})$ by edge bundles (Fig. 8c). We see that the bundle emerging from $\mathbf{q}$ precisely follows the branch, with darker edges near $\mathbf{q}$, and does *not* bifurcate to close branches to the right or left. This tells that points *along* the branch are nearer to $\mathbf{q}$ in D^n than points *across* the branch (which are not present in the bundles). Also, nearby branches have warm colors (low values of ρ_q^2 = near neighbors in D^2) while points along the branch containing point $\mathbf{q}$ are not. This confirms the message: across the branches we have a lot of *false neighbors*, which are near $\mathbf{q}$ in D^2 but are actually *intruders* to the projection neighborhood, since they are not part of $\nu_k^n(\mathbf{q})$. Hence, we conclude that points in the star branch are indeed much more similar to each other than points located in nearby branches. As such, the interpretation of this projection should be done differently than the intuitive "closer is better" one: points should be considered closer not based on their 2D distance, but based on their distance following the star branches, as if there were 'walls' between the branches.

Finally, we select a point $\mathbf{q}$ in the star center and draw its entire D^n neighbor-set $\nu_k^n(\mathbf{q})$ using edge bundles (Fig. 8d). The drawn bundles end all over the projection, both very close but also very far from $\mathbf{q}$. This tells us that the D^n neighbors of $\mathbf{q}$ are spread over large extents of the projection. To better assess this, we then color code the projection by the $\rho_q^2(\mathbf{q})$ ranks of $\mathbf{q}$'s 2D neighbors $\nu_k^2(\mathbf{q})$. The appearing red spot thus shows precisely the extent of the k-neighbors of $\mathbf{q}$ in 2D, *i.e.*, shows us the size of $\nu_k^2(\mathbf{q})$. As this extent is much *smaller* than the reach of the bundles, we conclude that, indeed, many of the D^n neighbors of the selected point $\mathbf{q}$ are placed very far away from it in the projection.

5. Discussion

We next discuss several technical aspects of our method.

Workflow: Key to the success of a visual analytics application is proposing a workflow that users should follow to obtain desired insights. In our case, this workflow has the following four steps: (1) Use the centrality view to determine a suitable value for k at which the neighborhood size matches well the size of the patterns of interest in the projection; (2) Use the set-difference view to find out how errors are spread over the projection and what is the average error size; (3) Use the sequence-difference view to locate outliers, *i.e.* zones having the largest (or smallest) errors; (4) Select points of interest in the projection, either in areas describing observations relevant for application-dependent tasks, or else in high-error areas, and use edge bundles to find where their true neighbors are located; (5) Use insights from (2-4) to determine where the true boundaries of strongly-related point groups in the projection are; (6) Decide, based on (2-5), whether the projection supports the tasks at hand in presence of all found errors, and how to interpret the projection; or whether these errors are too large and/or numerous, which means that a different projection is required.

Scalability: Our projection metrics require the computation of k neighborhoods for N points in D^n and D^2. We do this efficiently by using the fast nearest-neighbor search provided by [AMN*98], which is $O(kn \log N)$, and can handle any number of dimensions n and many types of distance metrics. Practically, this means that we can compute our metrics, and generate our views, in real time on a typical PC computer for tens of thousands of points having tens of dimensions. The approximation error of ANN is a limitation of its use, but the maximum error can be specified by the user, allowing a fine-grained control on the tradeoff between accuracy and running time.

6. Conclusions

We have presented a visual exploration method for finding and explaining neighborhood-preservation errors in multidimensional projections. Our method supports assessing the usefulness of a projection in terms of determining its overall quality, local errors, and how these errors should be considered when interpreting the projection to reason about the underlying high-dimensional data. Our techniques complement

and extend the set of existing tools for projection exploration including aggregate error metrics, neighborhood-preservation plots, and distance-error views, thereby offering users additional ways to reason about the usefulness and usability of multidimensional projections for data analysis tasks.

We plan to extend our work to support user-driven projection construction, by allowing users to interactively change the projection [ABD*12, PPM*15] while explicitly seeing how local neighborhood errors are created or diminished. Additionally, we aim to validate the effectiveness of our exploratory techniques by testing them on end-to-end use-cases involving additional datasets and projections.

Acknowledgements

This research was funded and supported by grant 2012/07722-9 of the São Paulo Research Foundation (FAPESP) and by CAPES–NUFFIC 028/11, an international research cooperation between the Brazilian Coordination for the Improvement of Higher Education Personnel (CAPES) and the Netherlands organization for international cooperation in higher education (NUFFIC).

References

[ABD*12] AMORIM E., BRAZIL E., DANIELS J., JOIA P., NONATO L., SOUSA M.: iLAMP: Exploring high-dimensional spacing through backward multidimensional projection. In *Proc. IEEE VAST* (2012), pp. 53–62. 8

[AMN*98] ARYA S., MOUNT D., NETANYAHU N., SILVERMAN R., WU A.: An optimal algorithm for approximate nearest neighbor searching. *J. of the ACM 45*, 6 (1998), 891–923. 7

[Aup07] AUPETIT M.: Visualizing distortions and recovering topology in continuous projection techniques. *Neurocomputing 10*, 7-9 (2007), 1304–1330. 1, 2

[BG05] BORG I., GROENEN P.: *Modern multidimensional scaling: Theory and applications*. Springer, 2005. 1, 2

[BP92] BAUER H., PAWELZIK K.: Quantifying the neighborhood preservation of self-organizing feature maps. *IEEE T Neural Networ 3*, 4 (1992), 570–579. 2

[BP07] BRANDES U., PICH C.: Eigensolver methods for progressive multidimensional scaling of large data. In *Proc. Graph Drawing* (2007), Springer, pp. 42–53. 2

[BTB13] BROEKSEMA B., TELEA A., BAUDEL T.: Visual analysis of multidimensional categorical data sets. *CGF 32*, 8 (2013), 158–169. 1

[dST04] DE SILVA V., TENENBAUM J.: *Sparse multidimensional scaling using landmark points*. Tech. rep., Stanford Univ., 2004. 2

[Ead84] EADES P. A.: A heuristic for graph drawing. *Congressus Numerantium 42* (1984), 149–160. 2

[EMdS*15] ETEMADPOUR R., MOTTA R., DE SOUZA J., MINGHIM R., DE OLIVEIRA F., CRISTINA M., LINSEN L.: Perception-based evaluation of projection methods for multidimensional data visualization. *IEEE TVCG 21*, 1 (2015), 81–94. 2

[FL95] FALOUTSOS C., LIN K.-I.: FastMap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proc. ACM SIGMOD Intl. Conf. on Management of Data* (1995), pp. 163–174. 2

[Git15] GITHUB: Online project hosting using git, 2015. 6

[GKN04] GANSNER E., KOREN Y., NORTH S.: Graph drawing by stress majorization. In *Proc. Graph Drawing* (2004), Springer, pp. 239–250. 2

[GZZ05] GENG X., ZHAN D.-C., ZHOU Z.-H.: Supervised nonlinear dimensionality reduction for visualization and classification. *IEEE Trans Syst Man Cybern 35*, 6 (2005), 1098–1107. 2

[HET12] HURTER C., ERSOY O., TELEA A.: Graph bundling by kernel density estimation. *CGF 31*, 3 (2012), 865–874. 5

[JPC*11] JOIA P., PAULOVICH F. V., COIMBRA D., CUMINATO J. A., NONATO L. G.: Local affine multidimensional projection. *IEEE TVCG 17* (2011), 2563–2571. 2, 3

[LA11] LESPINATS S., AUPETIT M.: CheckViz: Sanity check and topological clues for linear and non-linear mappings. *CGF 30*, 1 (2011), 113–125. 1

[Lic13] LICHMAN M.: UCI machine learning repository, 2013. 3

[LM06] LANZA M., MARINESCU R.: *Object-Oriented Metrics in Practice*. Springer, 2006. 6

[LW71] LEVANDOWSKY M., WINTER D.: Distance between sets. *Nature 234*, 5323 (1971), 34–35. 3

[LW03] LI J., WANG J. Z.: Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE TPAMI 25* (2003), 1075–1088. 6

[MCMT14] MARTINS R., COIMBRA D., MINGHIM R., TELEA A.: Visual analysis of dimensionality reduction quality for parameterized projections. *Comp. & Graph. 41* (2014), 26–42. 2, 3, 5

[MLN*13] MOTTA R., LOPES A., NOGUEIRA B., REZENDE S., JORGE M., DE OLIVEIRA M.: Comparing relational and non-relational algorithms for clustering propositional data. In *Proc. ACM Symp. on Applied Computing* (2013), pp. 150–155. 2

[MMLO15] MOTTA R., MINGHIM R., LOPES A., OLIVEIRA M.: Graph-based measures to assist user assessment of multidimensional projections. *Neurocomputing 150* (2015), 583–598. 2

[PNML08] PAULOVICH F. V., NONATO L. G., MINGHIM R., LEVKOWITZ H.: Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. *IEEE TVCG 14*, 3 (2008), 564–575. 1, 2, 3, 6

[PNSK*06] PANG-NING T., STEINBACH M., KUMAR V., ET AL.: *Introduction to data mining*. Pearson, 2006. 2

[PPM*15] PAGLIOSA P., PAULOVICH F. V., MINGHIM R., LEVKOWITZ H., NONATO L. G.: Projection inspector: Assessment and synthesis of multidimensional projections. *Neurocomputing 150* (2015), 599–610. 2, 8

[PSN10] PAULOVICH F. V., SILVA C., NONATO L. G.: Two-phase mapping for projecting massive data sets. *IEEE TVCG 16* (2010), 1281–1290. 1

[Sam69] SAMMON J. W.: A nonlinear mapping for data structure analysis. *ACM. Trans. Comp. C-18* (1969), 401–409. 2

[SvLB10] SCHRECK T., VON LANDESBERGER T., BREMM S.: Techniques for precision-based visual analysis of projected data. *Inf Vis 9*, 3 (2010), 181–193. 1, 2, 5, 6

[TMN03] TEJADA E., MINGHIM R., NONATO L. G.: On improved projection techniques to support visual exploration of multidimensional data sets. *Inf Vis 2*, 4 (2003), 218–231. 1, 2

[Tor65] TORGESON W. S.: Multidimensional scaling of similarity. *Psychometrika 30* (1965), 379–393. 1

[VDHM97] VILLMANN T., DER R., HERRMANN M., MARTINETZ T. M.: Topology preservation in self-organizing feature maps: exact definition and measurement. *IEEE T Neural Networ 8*, 2 (1997), 256–266. 1, 2

[vdMH08] VAN DER MAATEN L., HINTON G.: Visualizing data using t-SNE. *J Mach Learn Res 9* (2008), 2431–2456. 1, 2

[vdMH12] VAN DER MAATEN L., HINTON G.: Visualizing non-metric similarities in multiple maps. *Machine Learning 87*, 1 (2012), 33–35. 1, 2

[VK01] VENNA J., KASKI S.: Neighborhood preservation in nonlinear projection methods: An experimental study. In *Artificial Neural Networks/ICANN*. Springer, 2001, pp. 485–491. 2