Chapter 4

Visualizing Neighborhood Preservation

The visual analysis of projection errors introduced in Chapter 3 highlighted the importance of making it clear, for the user of a projection, where and how much the chosen projection method maintains aspects of the original structure of the data under investigation. We showed that errors are not always equally distributed over all points, especially when considering nonlinear projection techniques. As such, having a local insight into the quality of the projection provides the user with good advice for the decision of which regions should or should not be trusted. Interactive tools for fine-grained investigation are also crucial in the process of analysing errors, allowing the user to confirm or deny hypothesis that come from more general views.[1]

Chapter 3 also highlighted that the preservation of the structure of the data, in the context of a projection, can be further quantified in terms of several metrics. One of these, the preservation of Euclidean distances between point pairs, can be used to detect and analyse various quality aspects related to projections, such as missing and false neighbors, defined both at the level of individual observations and groups of observations.

However, distance-based quality metrics are not always the way to capture the desired quality aspects of a projection. One such situation is the widespread analysis task involving finding and reasoning about point groups and outliers. For such tasks, actual *distances* between points are less important than the *neighbors* of points. Indeed, we visually decide that a point-set in a projection 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. Similarly, we visually decide that a point is an outlier if it is located at a distance from all other points which is considerably larger than other inter-point distances in the same projection. In both above cases, groups and outliers can be reliably identified even if distances are not faithfully preserved by the used projection technique. The element that plays a key role here is the preservation of *neighborhoods* by the projection technique – or, in other words, the fact that the k nearest neighbors for a point in the projection are the same as the k nearest neighbors of the same point in the original high-dimensional space. As such, understanding neighborhood preservation errors is of a similar importance to understanding distance-preservation errors.

In this chapter 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. We introduce several visualizations that allow selecting suitable scales or levels-of-detail to examine such errors, and next show these errors and support users in understanding and using the projection in their presence. Our explanatory methods are simple to implement, computationally scalable,

[1] The content of this chapter is based on the paper *Explaining Neighborhood Preservation for Multidimensional Projections* (R. Martins, R. Minghim, A. Telea), Proc. Computer Graphics and Visual Computing (CGVC), eds. R. Borgo and C. Turkay, Eurographics, 2015.

apply to any projection technique, and can be easily integrated in classical scatterplot views of projections.

In this context, our main contributions are (1) three neighborhood preservation metrics that adapt [118, 198] 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.

4.1 Measuring and Visualizing Neighborhood Preservation

4.1.1 Preliminaries

Let $D^n = \{\mathbf{p}_i\}$ be a set of n-dimensional points, and let $D^m = \{\mathbf{q}_i\}$ the projection of D^n into a space having m dimensions. In this context, each high-dimensional point $\mathbf{p}_i$ has a unique corresponding low-dimensional point $\mathbf{q}_i$. For simplicity of notation, and when we need to refer solely to the identity of such a point, without specifying which of its two 'versions' $\mathbf{p}_i$ or $\mathbf{q}_i$ we consider, we will denote it simply by point i.

With the above convention, we define a k-neighborhood of a point i as the set

$$\nu_k(i) \subset \{1, \ldots, N\} \tag{4.1}$$

of the k-nearest neighbors of i (for a user-given k), sorted increasingly by Euclidean distance to i.

When a high-dimensional dataset D^n is projected into m dimensions, each point i has two k-neighborhoods: one in D^n, denoted by $\nu_k^n(i)$, and other in D^m, denoted by $\nu_k^m(i)$. All the examples in this chapter use 2D projections ($m = 2$) and Euclidean distances, for illustration simplicity. However, the presented techniques can be equally easily applied to 3D projections and/or other distance metrics.

Based on from these two k-neighborhoods, four types of points can be identified as important for the neighborhood preservation analysis of any point i, as illustrated by the Venn diagram in Fig. 4.1:

- **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 $\nu_k^n(i)$, are considered not very important by the projection method, and therefore are pushed outside $\nu_k^2(i)$. Missing neighbors are, thus, not found when visually examining the projection around $\mathbf{q}_i$.
- **false neighbors** $= \{\mathbf{p}_j \notin \nu_k^n(i) \wedge \mathbf{q}_j \in \nu_k^2(i)\}$
  These points are originally far away from $\mathbf{p}_i$ (outside $\nu_k^n(i)$), but are brought close to $\mathbf{q}_i$ in the resulting projection. False neighbors are, thus, points we visually see as being close to point $\mathbf{q}_i$ in the projection, but which are in reality far from point $\mathbf{p}_i$ in the high-dimensional space D^n.
- **true neighbors** $= \{\mathbf{p}_j \in \nu_k^n(i) \wedge \mathbf{q}_j \in \nu_k^2(i)\}$
  These are points which are close to both $\mathbf{q}_i$ and $\mathbf{p}_i$, so their visual representations are accurate regarding the k-neighborhood of i.

- **not neighbors** $= \{\mathbf{p}_j \notin \nu_k^n(i) \wedge \mathbf{q}_j \notin \nu_k^2(i)\}$
  These are points which are not near i in either D^n or D^m for a given value of k.

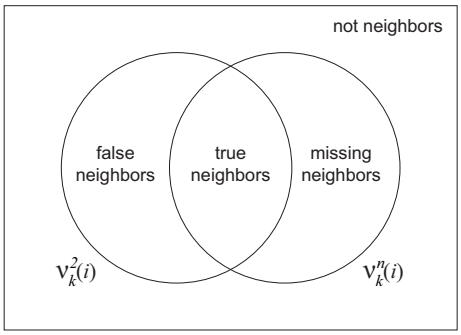

Figure 4.1: The four types of points that can be derived from the two k-neighborhoods $\nu_k^n(i)$ and $\nu_k^2(i)$ of a point i when analysing neighborhood preservation.

Considering that projections are used to reason about point neighborhoods in D^n by using point neighborhoods in D^2, we do not want a projection to create any false or missing neighbors, as these would mislead users when visually 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 analyse 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 [116]. 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 [95, 139, 118]. For the projection technique we use the well-known high-quality nonlinear LAMP technique [95]. As for the investigation of distance-based projection errors discussed in Chapter 3, other datasets and projection techniques can be directly used.

4.1.2 The Centrality Preservation view

Since our neighborhood preservation analysis method uses a fixed k to compute neighborhoods, one initial challenge is to understand, for the projection at hand, what is the effect of the value of k on the neighborhoods being analysed, or, in other words, to comprehend what would be a *good* value for k taking into account both the distribution of points and the specific goals of the analysis. Indeed, without having such a value, one would need to porentially analyse neighborhood preservation for all values $1 < k \leqslant N$

allowed by a dataset of N points. This is clearly prohibitive, both in computational costs and in user-effort terms.

Considering this problem, we introduce the *centrality preservation* metric

$$CP_k(j) = \sum_{1 \leqslant i \leqslant N, j \in \nu_k(i)} k - \rho_i(j) + 1, \tag{4.2}$$

for a set D of N points, where $\rho_i(j)$ is the *rank* of a k-neighbor j in $\nu_k(i)$ when sorted in ascending order by distance to i, so the nearest neighbor has $\rho_i = 1$ and the farthest neighbor has $\rho_i = k$.

The expected behavior of CP_k is that points j that are near neighbors to many other points i of D should be assigned high CP_k values, such as points which are *central* with respect to the structure of the set D, while points close to the *periphery* of D, which are not near neighbors to many other points, should be assigned low CP_k 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. For the detailed description of these techniques, we refer to Sec. 3.2.2, where we introduced them first for the visualization of distance preservation errors. A first example of the result is shown in Fig. 4.2, for CP_k computed in both D^2 and D^n.

Let us first consider CP_k^2, *i.e.* the *centrality preservation* computed over the D^2 projection space (Figs. 4.2a-c). As expected, in the D^2 space, points located near central areas have higher values (closer to red) than points located near peripheral areas, which have lower values (closer to blue). However, the exact distribution of these values throughout the projection varies considerably as the k parameter changes. Fig. 4.2a shows that with too low k values CP_k highlights very small changes in local point density. Assessing neighborhood preservation at such scales might not be interesting for most real-world scenarios – even a tiny shift in the points' positions would create different CP_k^2 values. Conversely, setting too large k values (Fig. 4.2c) highlights too coarse-scale patterns that do not match the shape of the projection, since points j are being considered as k-neighbors even when they are very far from the reference point i. In the limit case $k = N$, CP_k^2 will show a single circular gradient covering the entire projection. This highlights the fact that, for this extreme k value, all points $j \neq i$ are considered to be neighbors of any reference point i – a configuration which is arguably not useful for any practical analysis of neighborhood preservation. In-between values, *e.g.* $k = 180$ for our running example (Fig. 4.2b), highlight centrality patterns which match well the perceived *shape* of the projection — we see how red bumps nicely match the main point-groups visible in the projection.

Hence, visualizing CP_k^2 helps finding a good scale at which to assess neighborhoods, and the D^2 centrality view can be used to select a k which best matches the desired level-of-detail to explore the projection, based on the match between the shapes we see in the projection and the CP_k^2 peaks. In detail, we aim to set k to obtain roughly one such peak per individual set of points in the projection which the user regards as forming a separate group. Note that this is not always possible: For the dataset in Fig. 4.2, the closest we can come to this configuration is given by setting $k = 180$, yielding the image in Fig. 4.2b. We see here how the left and bottom-right groups of points in the projection

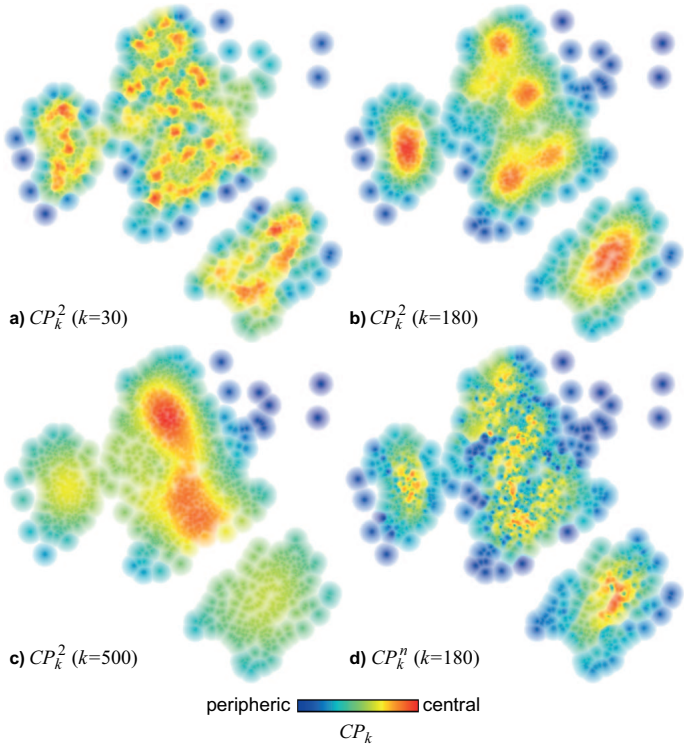

Figure 4.2: *Centrality preservation* view, *segmentation* dataset, LAMP projection. (a-c) Centrality CP_k^2, for three neighborhood sizes k. (d) Centrality CP_k^n, for $k = 180$ neighbors.

are each nicely matched by a single red peak. However, the large central point group is covered by several such peaks.

Once the k value is set, the analysis can now turn to the original D^n neighborhoods by changing the values drawn over the projected points D^2 to CP_k^n. To explain this design, consider a projection that perfectly preserves neighbors: In such a case, $CP_k^n(i) = CP_k^2(i), \forall i \in \{1, \ldots, N\}, \forall k \in \{1, \ldots, N\}$, and the visualization of CP_k^n should match the already-understood color gradient 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. 4.2d shows CP_k^n for $k = 180$, the same value of k as Fig. 4.2b (for CP_k^2). While the trend of red central points and blue peripheral points is somewhat similar in Figs. 4.2b and 4.2d, the smooth red-to-blue gradient in Fig. 4.2b is partially lost in most areas of the image. This is an indication that there are neighborhood preservation errors *and* that they are spread out all over the projection. By itself, however, this view is not detailed enough to clearly point to the user where the errors are occurring, what

kinds of errors these are, and why they are happening. To answer these questions, we refine the exploration of neighborhood preservation with the set-difference and sequence-difference views presented next in Sections 4.1.3 and 4.1.4 respectively.

4.1.3 The Set Difference view

The second proposed view, called the *set difference* view, compares the neighborhoods $\nu_k^n(i)$ and $\nu_k^2(i)$ of each data point i using the Jaccard set-distance [112], by computing

$$JD_k(i) = 1 - \frac{\nu_k^2(i) \cap \nu_k^n(i)}{\nu_k^2(i) \cup \nu_k^n(i)}. \tag{4.3}$$

This value represents the neighborhood preservation *error* of each point i. Its interpretation is simple: $JD_k(i) = 1$ means that the D^n neighborhood of point i was completely lost by the projection, while $JD_k(i) = 0$ means that the projection preserved the k neighbors of i perfectly. However, the neighbors' ranks, positions, and distances relative to point i are not considered by this error metric.

Fig. 4.3 shows three set-difference views for the same k values as in Fig. 4.2, for ease of comparison. According to the interpretation of JD_k, high values (warm colors) show poor neighborhood preservation while low values (cold colors) show areas where neighbors are well preserved. With these new images it is now possible to get extra insights into the neighborhood preservation of the projection, in addition to the information provided previously by the *centrality preservation* view (Fig. 4.2).

First, for the fine-grained scale $k = 30$ (Fig. 4.3a), 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. 4.3b), determined in Sec. 4.1.2 to be a *good* level-of-detail to examine this dataset, we see that both smaller point-groups B_1 and B_2 have very good neighborhood preservation (low JD_k values). This confirms that the LAMP method was right to separate them from the central group A.

At the same scale ($k = 180$), an *isthmus* (Fig. 4.3b, marker C) can be identified 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. 4.1.4).

Several red islands can also be found in the central group A (Fig. 4.3b, zones D). Increasing k to 500, these islands are reduced to a few outliers (Fig. 4.3c, zones D), which suggests that, on a coarse scale (a scale that would better match its size, since it is a larger group than the others), group A has little neighborhood preservation issues, so it is *indeed* a large group in D^n space. However, the relatively isolated group F remains red even at a coarse scale; this indicates that, no matter the value of k, these points are wrongly projected 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, a user may think of it as simply a compact large cluster of very similar points. Yet, in the *set difference* view,

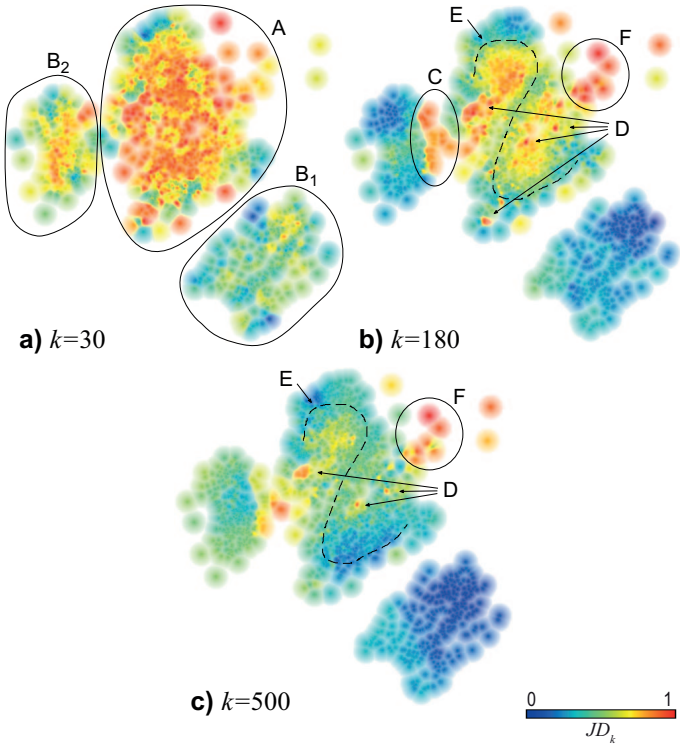

Figure 4.3: *Set difference* view, *segmentation* dataset, LAMP projection. The figure uses the same k values as in Figs. 4.2a-c.

we see a Z-shaped *corridor* of points of medium error (Fig. 4.3b, marker E) that winds through the high-error red islands inside group A. This suggests that group A may not be that homogeneous in D^n space. We will explore this finding next using our subsequent views.

4.1.4 The Sequence Difference view

While the JD_k metric (Eq. 4.3) comes naturally from the problem of comparing two possibly-overlapping sets of points, and its accompanying *set difference* view (Sec. 4.1.3) shows an easy-to-interpret picture of how many **true neighbors** a projected point has, it has limitations that are worth noticing. As can be observed in Fig. 4.3, for example, the results are quite sensitive to the setting of k. The three pictures generated by increasing the neighborhood size are considerably different from each other; areas where high-errors were abundant in one picture can be seen with very low errors in the others, and high-error outliers are also not constant. Since, as outlined earlier, k is a free parameter that different users may set differently for the same projection, having a view that is strongly affected by the setting of k can be a threat to the consistency of the results obtained from the analysis process.

We argue that this perceived limitation is related to two properties of JD_k: (a) it ignores changes in the structure of the k-neighborhoods other than the inclusion or exclusion of neighbors, such as whether and how much a k-neighbor j changes *ranks* (in the sense introduced in Eqn. 4.2) when comparing both k-neighborhoods; and (b) how *important* this k-neighbor j is, in terms of how much it is near the reference point i.

The work of Pagliosa *et al.* [135] proposes a *Smooth Neighborhood Preservation* metric that deals with these problems by using the distances between neighbors as weights for the error, so that the more distant a neighbor is, the less impact it has on the error. Alternatively, to decrease the analysis sensitiveness to the setting of k and also to account for the importance of neighbors, using their ranks instead of distances, we propose next a *sequence difference* metric to compare $\nu_k^2(i)$ and $\nu_k^n(i)$ as

$$\begin{aligned} SD_k(i) = & \sum_{j \in \nu_k^2(i)} (k - \rho_i^2(j) + 1) \cdot |\rho_i^2(j) - \rho_i^n(j)| \\ & + \sum_{j \in \nu_k^n(i)} (k - \rho_i^n(j) + 1) \cdot |\rho_i^2(j) - \rho_i^n(j)|, \end{aligned} \tag{4.4}$$

where $\rho_i(j)$ is the *rank* of a k-neighbor j in $\nu_k(i)$, as defined in Eqn. 4.2. The first term in each sum in Eqn. 4.4 assigns a higher weight to the displacement of nearer (lower-rank) neighbors, capturing the assumption that not preserving the rank of a near neighbor is worse, in terms of interpretation of the resulting projection, than not preserving the rank of a distant neighbor. This is based on the way users typically interpret a projection visually, *i.e.* by locally scanning and querying small point neighborhoods to find what is most similar to a given point, before they scan further points. The metric's second term in each sum penalizes neighbors j which do not keep ranks after projection, *i.e.* $\rho_i^2(j) \neq \rho_i^n(j)$, by how much their rank is changed.

Figure 4.4 shows the *sequence difference* view for our running example. The four pictures, generated for increasing scales of k, show much less differences and are more stable when compared to the earlier views (Figs. 4.2 and 4.3), as expected. All medium and high-error areas exposed by the *set difference* view (Fig. 4.3b) can still be seen, and high-error outliers (small red dots marked in Fig. 4.4) are now much better visible at all scales.

To explain this effect, one might think of SD_k (Eqn. 4.4) as a rank-weighted version of JD_k (Eqn. 4.3). 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.

One final observation about the definition of SD_k relates to how it handles **missing** and **false neighbors** (as defined in Sec. 4.1.1). A missing neighbor j that was originally part of $\nu_k^n(i)$ but was pushed outside of $\nu_k^2(i)$ has a ρ_i^2 that is greater than k. The absolute-difference terms $|\rho_i^2(j) - \rho_i^n(j)|$ of each sum in the definition of SD_k (Eqn. 4.4)

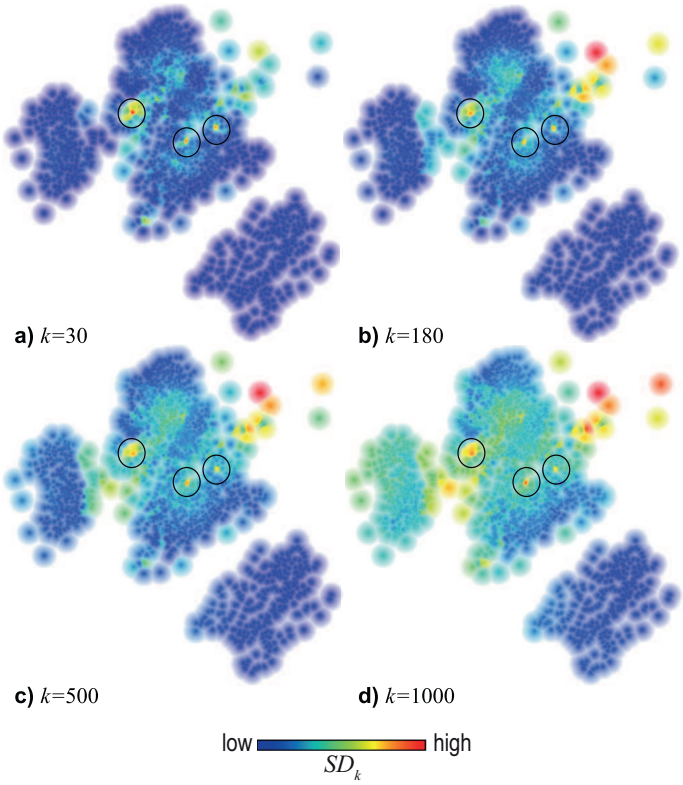

Figure 4.4: *Sequence difference* view, *segmentation* dataset, LAMP projection, for four increasing scales (k values).

always represent how much the rank of a point changed between both spaces, no matter the k. The same happens to false neighbors. In a way, this approach can be considered to join the analysis of neighborhood preservation with the analysis of distance preservation, since SD_k is able to 'see' beyond the fixed k value. However, it must be noted that (a) the metric considers only neighbors (whether true, false or missing), so the analysis is still restricted by the value of k; and (b) our metric only considers ranks and not actual Euclidean distances between points. This last property is important because it allows using this metric for the analysis of datasets for which the original distances are not directly comparable to distances in the low-dimensional projection space D^2. This topic is further detailed in Chapter 7.

4.1.5 Refining the exploration

Using the *set* and *sequence difference* views, we are able to notice a few different areas of the projection from our running example that show different levels of neighborhood-preservation errors. This is by itself an important insight into the overall quality of the projection, but not a fully detailed one: By using these views, a user cannot distinguish

between errors that are due to false neighbors and errors that are due to missing neighbors. Additionally, it is not only important to know the types of occurring errors, but also where do these errors come from. For instance, if we have a case of false or missing neighbors, important questions to address are which are these false neighbors, and where are the missing neighbors, respectively. To address these tasks, we next propose visual representations of $\nu_k^n(i)$ and $\nu_k^2(i)$ for specific selected $\mathbf{q}_i$'s, similar to the techniques used in Sec. 3.2.4 for the detailed analysis of distance-preservation errors.

The first scenario of local analysis is shown in Fig. 4.5 for our running example. The proposed visual encoding works as follows: For any selected point $\mathbf{q}_i$, its **true k-neighbors** are colored according to their proximity to i in D^n, i.e. $k - \rho_i^n + 1$, so that higher values mean nearer k-neighbors. The colormap for this analysis, while very similar to the ones used in previous figures in this chapter, has one distinct feature: Transparency is assigned to points according to their values, increasing linearly with the proximity to i (the same value mapped to color). Hence, nearer k-neighbors are more opaque, while farther k-neighbors are slightly less apparent. All points that are **not k-neighbors** – they are neither in $\nu_k^n(i)$ nor in $\nu_k^2(i)$, so they are not relevant to this analysis – are assigned $\rho_i^n = k + 1$, as if they were all just outside the k-neighborhood. This means that in the view they are assigned the value 0 and, thus, get the maximum assigned transparency. This design is useful for a better visibility of missing neighbors. These are visualized by connecting them with lines to the reference point i, and next bundling these lines to reduce clutter caused by many crossing lines and also emphasize the main groups of missing neighbors, similarly to the missing neighbors finder technique proposed in Sec. 3.2.5 for distance-based errors. The bundled lines are colored using a gray scale where closer k-neighbors are darker than farther ones.

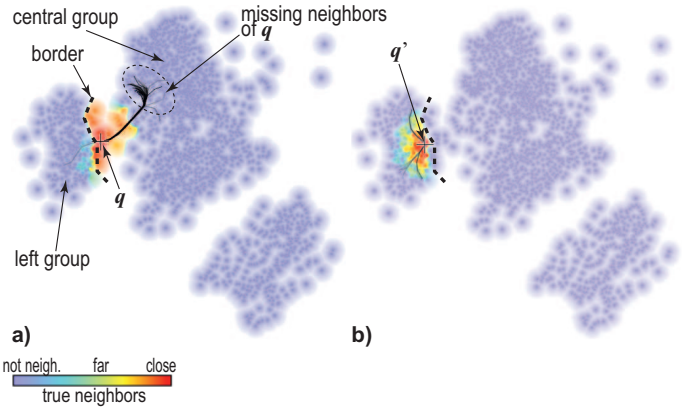

Figure 4.5: Local analysis of the connection between the left and central groups. The visual border, seen also in Figs. 4.3b and 4.4, is marked by a dotted curve.

The set difference view (Fig. 4.3b) shows a salient border, or color gradient, that seems to separate the highly-coherent group of points in the left of the projection from the large central group via an isthmus of in-between points (see Fig. 4.3b, marker C).

This finding suggests the hypothesis that the left point-group is actually well separated from the central group in high-dimensional space, and that the linking isthmus is mainly an artifact of the projection.

To verify this hypothesis, we first select the point $\mathbf{q}$ (Fig. 4.5a), to the right of the border and inside the high-error area of the isthmus. Warm colors spread only to the right of the border, which means the nearest **true k-neighbors** of $\mathbf{q}$ are all placed to the right of the border and towards the large cluster. With the **missing k-neighbors** (bundled edges) it is exactly the same: dark edges all point to the inside of the central group, which means that k-neighbors that were originally close to $\mathbf{q}$ in D^n are located in the inner parts of the central group. In other words, we conclude that the points on the isthmus form indeed a tight k-neighborhood in D^n, as suggested by the previous views, but while that k-neighborhood is visually close to the left group in D^2, there are no strong indicators that it is also the case in D^n.

Neighborhood relationships are not commutative, however: If a point A is one of the nearest k-neighbors of a point B, that does not necessarily mean that B is also one of the nearest k-neighbors of A (and vice-versa, for far neighbors). Therefore, to conclude the investigation of our hypothesis, it is also important to analyse the points outside of the isthmus and in the left group. Repeating the same operations for a point $\mathbf{q}'$ located to the *left* of the border (and outside the high-error area of the isthmus), we can see a nearly exactly complementary picture (Fig. 4.5b): All true and missing k-neighbors of $\mathbf{q}'$ are located inside the left group. The only difference is that $\mathbf{q}'$ has fewer **missing k-neighbors**, and the ones which are there are not important – light-gray edges mean that these k-neighbors were originally already far from $\mathbf{p}_i$ in D^n, and they do not reach far into the left group, but stop at the border of the colored true k-neighbors.

Together, the above two findings lead to the conclusion that the border we discovered by our views, highlighted in Fig. 4.5 as a dotted curve, is indeed an important and well-defined division between the multidimensional data points from the *left* and the *central* groups. Without the tools and views presented in this section, starting all the way from the general detection of a potentially problematic region and ending in the specific and detailed analysis of the k-neighborhoods of the points around the area, this insight might not be easily obtained by a user of this projection. Further on, without this insight, the visual analysis of this complex area in the projection might mislead and erroneously drive the exploration of the data.

4.1.6 Ground truth analysis and comparison

The *segmentation* dataset we used as a running example is a collection of 2100 3x3 pixel regions, also called instances, drawn randomly from a database of 7 outdoor images and represented by attributes related mainly to the positions and colors of the pixels. The information of which image each instance comes from is available as a class attribute and can be used to separate the instances in 7 different groups. However, this information is not used by the projection method, in line with typical machine learning procedures where one wants to (a) automatically extract features from a dataset and next (b) validate the usage of these features by correlating them with manual annotations, or ground truth, information. In our context, assessing the projection in terms of finding

correlations of the neighborhood-preservation error and mix of class attributes is a potentially useful way to reason about the causes of the appearance of these errors.

Following this idea, we show our running example dataset in Fig. 4.6 colored by the ground truth (class attribute). Next, we analyse how much of the patterns and visual features related to neighborhood preservation, which we identified earlier with our proposed views, match the distribution and position of patterns implied by the class attribute. For clasity, we show the color-coded projection image twice, once without and next with superimposed annotations. We also note that the Shepard attribute interpolation used earlier is now not employed, since the class attribute is a categorical, rathern than quantitative, one.

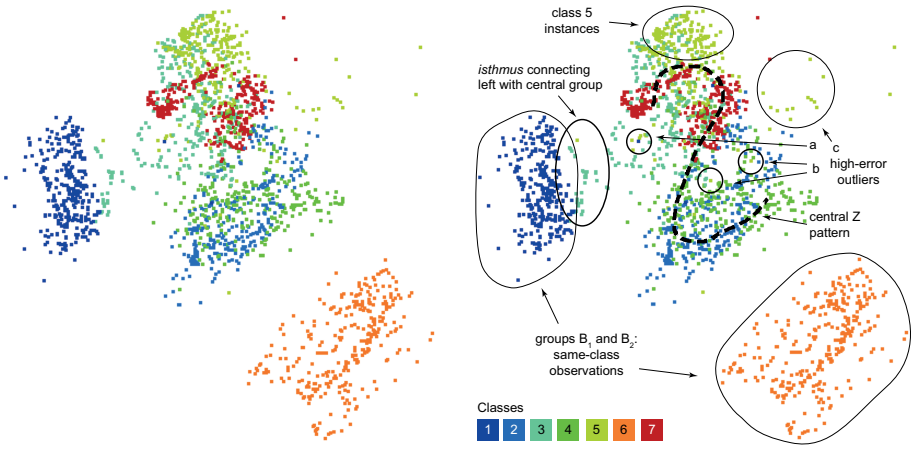

Figure 4.6: Original classification (*ground truth*) for the *segmentation* dataset.

Comparing Figure 4.6 with our earlier images showing neighborhood-preservation errors (Figs. 4.3 and 4.4), several observations can be made: Classes 1 and 6 are very well separated in the projection, and their respective areas coincide with high neighborhood-preservation areas in the projection. High-error zones detected earlier inside the large central group are also apparent when comparing the mixing of classes; one point (outlier **a**), part of the highly-cohesive class 1, can be found in the middle of points from classes 3 and 5, while a few points from class 5 (outliers **b** and **c**) are far away from the main zone covered by their class, in the upper part of the projection, so most of their nearer neighbors are from other classes. The Z-pattern inside the large central group, found in our previous views to show less errors than its surroundings, can now be traced roughly to the layout of points from classes 2 and 7 along the large cluster. Points in this Z-pattern, while still considerably mixed with points from other classes, have less error because their classes are relatively well-grouped among the clutter (so their near neighbors are mixed, instead of mostly good or mostly bad). Finally, our main focus area in Sec. 4.1.5, the isthmus connecting group B_2 to group A, can also be explained by observing the distribution of points from class 3, starting from the border of group B_2 and moving towards the inner area of group A.

In Section 2.4.2, a few existing techniques for the visual analysis of neighborhood preservation in projections were presented. These techniques share some of the goals of the views proposed in this chapter, but propose different approaches to reach them. To better illustrate the differences and added-value of our proposed views, we selected the work of Schreck *et al.* [159] for comparison, and computed the herein proposed *projection precision score (pps)* for the same example and k values as in Fig. 4.4.

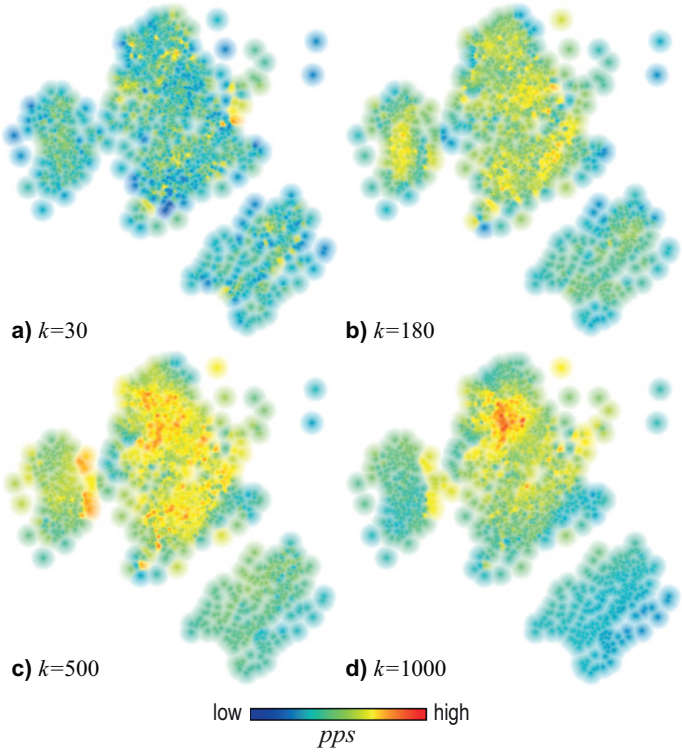

Figure 4.7: Projection precision score (*pps*) [159], *segmentation* dataset, LAMP projection, for four scales (k values).

The resulting visualization of the *pps* score is shown in Fig. 4.7. For the first two k values (30 and 180), it is hard to see any quality difference between different projection regions. For the last two k values (500,1000), some of the patterns we obtained with our proposed views show up 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. 4.3) are not visible. Last but not least, the technique proposed in [159] only proposes the visualization of the *pps* score, but does not introduce additional explanatory tools to refine the exploration by *e.g.* explaining the causes of

identified high-error patterns.

4.1.7 Additional examples

To expand on the initial examples from previous sections, we next offer additional insight into the way that our proposed neighborhood-preservation exploration tools work by showing how they can be useful in the analysis of different datasets and projections.

The first example discussed next uses the *Corel* dataset, consisting of 1000 instances where each represents a photograph described by 150 SIFT features, common in image analysis [115]. We constructed the projection of the data using LAMP [95] and obtained the star-shaped image in Fig. 4.8a. By observing the distribution of points in this projection, it can be hypothesized that there are several well-separated image clusters in the dataset, one for each branch, while a group of "average" images which have no separate group identity and/or which are similar to instances from many different groups populate the center of the projection.

We start by checking the centrality preservation for both D^n and D^2 (Fig. 4.8b–c). The selected value of $k = 75$ roughly captures the two-dimensional layout of the points into branches (Fig. 4.8b). The picture for the centrality in D^n is completely different (Fig. 4.8c): Most of the centrality seems to have been 'lost in the translation'. This does not mean (yet) that the projection is wrong in any specific way; it may be the case that the D^n space is so sparse that, for the selected scale k, no points stand out as being particularly central. However, this image does indicate that the projection layout differs from the original in terms of centrality. As such, it is interesting to investigate the neighborhood preservation to get more insight into why this loss of centrality happened.

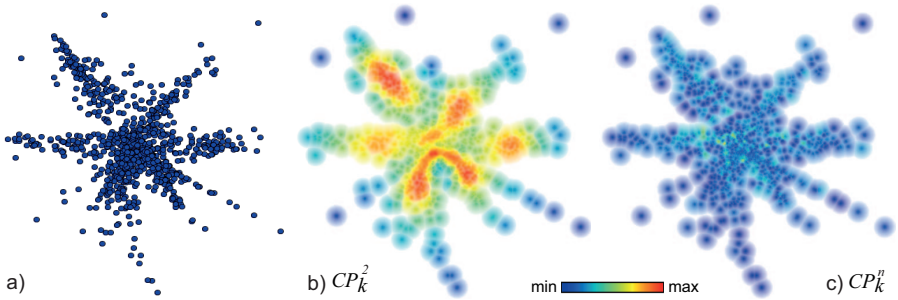

Figure 4.8: Corel dataset, LAMP projection, $k = 75$ neighbors. (a) Projection without error metrics. (b) Projection colored by CP_k^2. (c) Projection colored by CP_k^n.

We next inspect the set and sequence difference views (Fig. 4.8a,b). Similar to our previous running example, the set difference view shows a medium-to-high neighborhood preservation error spread rather uniformly over the projection (Fig. 4.8a), with a hint of some branches having better preservation than others. This difference between the branches is confirmed by the sequence difference view, which allows us to better distinguish the high-error and the low-error branches (Fig. 4.8b). Another advantage

of the sequence over the set difference view, in this case, is that it allows us to better comprehend the distribution of errors in the central area: While these errors are quite similar for the set difference view (Fig. 4.9a), the sequence difference view shows us more clearly the outliers with the largest errors (Fig. 4.8b). These insights indicate that some branches represent more cohesive groups, *i.e.* groups having more similar images, than other branches.

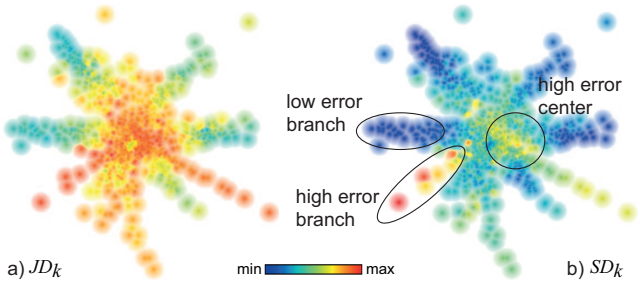

Figure 4.9: Corel dataset, LAMP projection, $k = 75$ neighbors. (a) Set difference view. (b) Sequence difference view.

Considering this difference of behavior in each branch, we move into investigating what are the relationships between branches or, in other words, whether D^2 neighbors *between* star branches represent the D^n neighborhood as well as D^2 neighbors *along* star branches. To highlight specifically the neighborhood preservation errors of a selected point $\mathbf{q}_i$, we slightly change the visual encoding used in previous figures of local analysis, to obtain the result showin in Fig. 4.10: Here, color encodes the false k-neighbors of $\mathbf{q}_i$ *inversely* by their rank, with values computed by $k - \rho_i^2 + 1$, so nearer false k-neighbors are shown with warmer colors while distance false neighbors have cold colors. This design is chosen to better highlight the more important problems – false k-neighbors that are nearer to $\mathbf{q}_i$ are worse than those which are farther. Points which are not k-neighbors are half-transparent blue, to make them less salient in the analysis. The encoding of the edges connecting a selected point with its neighbors is also slightly changed: Instead of showing **missing k-neighbors** only, edges in Fig. 4.10 show the entire neighborhood $\nu_k^n(i)$ of the selected point i.

With this visual encoding, we first select a point $\mathbf{q}_i$ on a star branch having low neighborhood preservation error (Fig. 4.10a). Seeing warm colors all around the point $\mathbf{q}_i$, especially in nearby branches, is an indication that this point is surrounded by false k-neighbors. These points, although near $\mathbf{q}_i$ in D^2, are actually 'intruders' in the projected neighborhood, since they are not part of $\nu_k^n(i)$. Edges emerging from $\mathbf{q}_i$ precisely follow the star branch that point i is located into and do not bifurcate to nearby branches to the right or left. This confirms our observation that points *along* the branch are nearer to $\mathbf{p}_i$ in D^n, and points *across* the branch (which are not present in the edge bundles) are not. However, these edges continue to reach for k-neighbors that are very far from the selected point, going all the way to a branch on the opposite side of the projection. This not only helps to explain the high errors detected in this area, but also

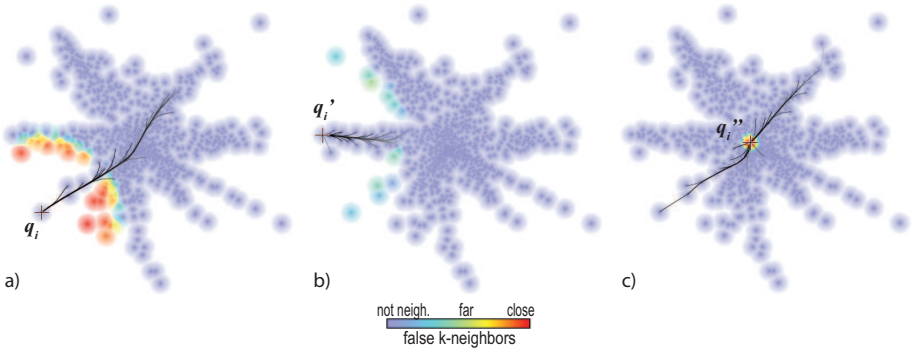

Figure 4.10: Local neighborhood analysis for the Corel dataset, LAMP projection, $k = 75$ neighbors.

show *where* these points should have been positioned to be nearer their D^n k-neighbors.

We next select a different point $\mathbf{q}_i'$ in another star branch. The resulting visualization shows a completely different situation (Fig. 4.10b). Not only are the false k-neighbors few and having cold colors, but the edges linking the selected point to its neighbors are all limited to its own branch, explaining why it presented such low errors values. Hence, we conclude that points in this star branch are indeed much more similar to each other than points located in nearby branches. As such, we argue that the interpretation of this projection should be done differently than the intuitive paradigm of *closer is better*, common when there are no insights into the projection's error: points should be considered closer not based only on their D^2 distance between each other, but specially based on their distance following the star branches, as if there were *walls* separating the branches.

Finally, we select a point $\mathbf{q}_i''$ in the star center (Fig. 4.10c). The edge bundles end up reaching both very close but also very far from $\mathbf{q}_i''$, into two opposite star branches. This confirms to us that, indeed, a point in the central region has a confusing k-neighborhood, with its D^n k-neighbors spread over large extents of the projection. Moreover, all its nearest k-neighbors in the projection, shown by the color coding, are actually **false**, which contributes even more to the high error values observed in this area.

The second application uses the *Github* dataset, a collection of 725 observations, each describing one among the highest-ranked open source software projects hosted at GitHub [65]. For each project, 30 software metrics were extracted describing various aspects of its development, such as size (lines of code, file count), average coupling and cohesion of modules, complexity, and number of forks and open issues [107]. We then create a visual representation of this 30-dimensional dataset using the LSP projection [139], with the goal of finding clusters of projects with similar quality attributes or trends in the spatial distribution that could indicate how these metrics behave in real projects. The result, presented in Fig. 4.11, shows mainly a large central cluster, surrounded by a few outliers. By looking at this projection without any of the visual helpers we introduced, a first question is raised: Are all projects indeed simply grouped into one similar set, or is the lack of separated clusters an artifact of the LSP technique?

To answer this, we first select a suitable neighborhood size (k value), as explained

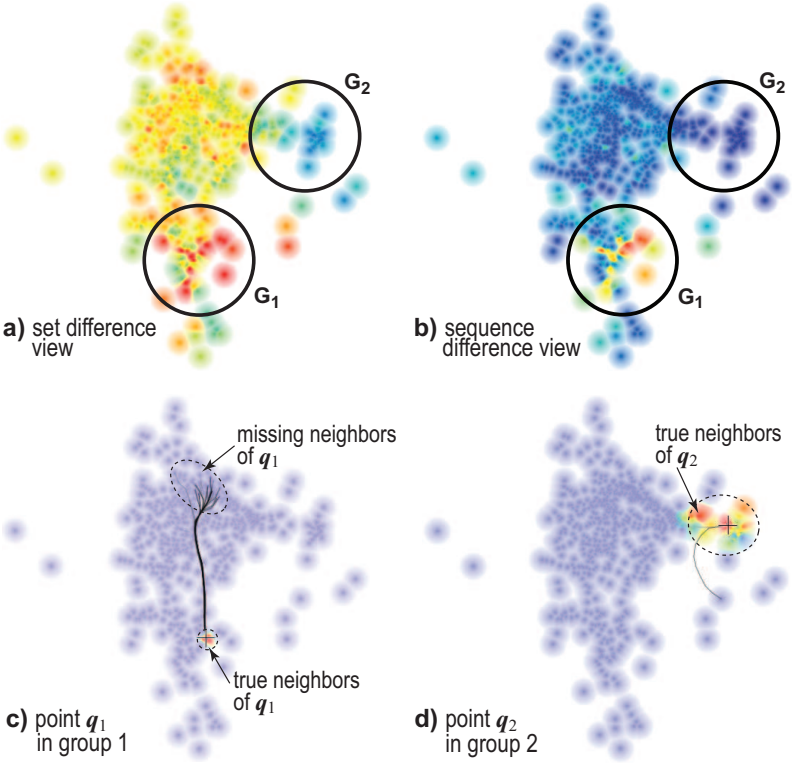

Figure 4.11: Neighborhood-preservation analysis for the *Github* dataset, LSP projection, $k = 72$ neighbors.

in Sec. 4.1.2, and obtain a value of $k = 72$. We start by inspecting the set difference view (Fig. 4.11a). While a quite poor neighborhood preservation can be observed for most points – which would suggest that LSP does not work optimally for this dataset – two groups of points seem to stand out for opposite reasons: G_1 stands out for its especially large errors, and G_2 stands out for showing lower errors than the remainder of the projection. The sequence difference view (Fig. 4.11b) confirms that G_1 has indeed the largest neighborhood preservation error in the projection, while G_2 still maintains very low errors. We have now two indicators of special point groups that, for different reasons, stand out from the initially roughly plain visualization.

We next examine these groups in detail by selecting a point of interest. The visual encoding is the same used in Sec. 4.1.5: Colors represent the selected point's true k-neighbors and the missing neighbors are shown with bundled edges. When we select a point $\mathbf{q}_1 \in G_1$ (Fig. 4.11c), we quickly see that $\mathbf{q}_1$ is surrounded by false k-neighbors – there is only a very small hotspot around the selected point that represents its true k-neighbors. Another interesting observation is that not only most of the missing k-neighbors are far away from the selected point, but all edges are high-valued (dark), which means that these neighbors were actually very close to $\mathbf{p}_1$ in D^n. These missing

neighbors are all grouped into a small area on top of the projection, which by all indications seems to be the right neighborhood for $\mathbf{q_1}$. The reason why these neighbors were placed so far away from $\mathbf{q}_1$ is, however, not something our techniques can explain – possible causes can range from the limitations of the LSP technique to the inappropriate tuning of the technique's parameters and to the similarity of these missing neighbors to other points located in the top area of the projection. In contrast, when selecting a point $\mathbf{q}_2 \in G_2$, we see almost no edges reaching out (the only out-reaching edge is light-gray and going to a nearby point), and the points around $\mathbf{q}_2$ are warm-colored, so most points in the projection around $\mathbf{q}_2$ are indeed its true neighbors (Fig. 4.11d). Hence, there are strong indications that G_2 is indeed a cohesive group in D^n and, since it is relatively well separated from the central group, we conclude that it represents a set of software projects that are highly-similar between themselves *and* quite different from the rest of the analysed set.

4.2 Discussion

We next discuss several technical aspects of our proposed views for analysing neighborhood-preservation.

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.

Generality: Our techniques can be applied for any projection technique, including linear [186, 24] and non-linear ones [139, 192, 179, 141], 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.

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 [6], 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.

Evaluation: Analysing neighborhood-preservation of projections is certainly not a problem with a single simple solution, given the variability and wide range of projection techniques, parameter settings, and datasets. The results presented in this chapter show how our proposed techniques work with a few representative and well-understood combinations of projections, parameter settings and datasets, which were selected to illustrate how our techniques behave in a few different key scenarios. However, we acknowledge that the presented experiments could not cover the entire aforementioned space of possibilities. As such, more strict and more thorough evaluations and validations are required. One important example of future work in this direction is the evaluation of the proposed metrics with the use of artificial datasets specifically crafted to evaluate possible limitations, such as their sensitivity to outliers.

Neighborhood *vs* Distance Preservation: As discussed in Sec. 3.4, distance-preservation and neighborhood-preservation are related, yet different, quality aspects of a multidimensional projection. Ideally, a projection should preserve both distances and neighborhoods. However, as seen in the many examples presented both in this chapter and Chapter 3, even state-of-the-art projections have challenges in meeting both these aspects. In such situations, we believe that selecting between using distance-preservation and neighborhood-preservation exploratory tools should be based mainly on the tasks implied by the application at hand: When these tasks involve comparing distances between points, then distance-preservation errors are clearly to be considered and explored. In contrast, when tasks involve reasoning about apparent groups of close points in the projection, then neighborhood-preservation errors should be explored first and foremost.

4.3 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. In particular, our neighborhood-preservation exploratory tools add themselves to the distance-preservation exploratory tools presented in Chapter 3: They propose a similar top-down analysis of the distribution and magnitude of errors, and employ similar visual interactive techniques to depict the measured errors. The two sets of exploratory tools complement each other, in the sense of offering the user detailed insight in projection errors that affect different types of exploratory tasks.

At a global level, both distance-preservation and neighborhood-preservation ex-

ploratory tools serve a number of different tasks, such as assessing the suitability of a given projection result for a given analysis goal; finding potential interpretation problems in a projection; and comparing several projections or projection techniques to decide which is more suitable, in terms of error, for a given analysis. The last point involves not only comparing different projection techniques against each other, but also comparing projections that create results of different dimensionalities, such as 2D or 3D scatterplots. The topic of comparing 2D and 3D projections from the perspective of produced errors, and next augmenting 3D projections with suitable explanatory mechanisms to bring them to a level of ease-of-use comparative to 2D projections, is explored in the next chapter.