# Explanatory visualization of multidimensional projections

*Rafael Messias Martins*

**Rafael Messias Martins**

# Explanatory visualization of multidimensional projections

Thesis submitted to the Instituto de Ciências Matemáticas e de Computação - ICMC-USP and to the University of Groningen - RUG, in partial fulfillment of the requirements for the degrees of the Doctorate Program in Computer Science and Computational Mathematics (ICMC-USP) and of PhD (RUG), in accordance with the international academic agreement for PhD double degree signed between ICMC-USP and RUG. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics / Computer Science

Advisor: Profa. Dra. Rosane Minghim (ICMC-USP, Brasil)
Advisor: Prof. Dr. Alexandru Cristian Telea (RUG, Holanda)

**USP – São Carlos**
**March 2016**

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

```
        Martins, Rafael Messias
M379e      Explanatory Visualization of Multidimensional
        Projections / Rafael Messias Martins; orientadora
        Rosane Minghim; co-orientador Alexandru C. Telea. --
         São Carlos, 2016.
           204 p.


           Tese (Doutorado - Programa de Pós-Graduação em
        Ciências de Computação e Matemática Computacional) --
         Instituto de Ciências Matemáticas e de Computação,
        Universidade de São Paulo, 2016.


           1. Computação Gráfica. 2. Visualização. 3. Análise
        Visual. 4. Dados Multidimensionais. I. Minghim,
        Rosane, orient. II. Telea, Alexandru C., co-orient.
        III. Título.
```

**Rafael Messias Martins**

# Visualização explanatória de projeções multidimensionais

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP e à Universidade de Groningen - RUG, como parte dos requisitos para obtenção dos títulos de Doutor em Ciências - Ciências de Computação e Matemática Computacional (ICMC-USP) e PhD (RUG), de acordo com o convênio acadêmico internacional para dupla titulação de doutorado assinado entre o ICMC-USP e a RUG. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional / Ciência da Computação

Orientadora: Profa. Dra. Rosane Minghim (ICMC-USP, Brasil)
Orientador: Prof. Dr. Alexandru Cristian Telea (RUG, Holanda)

**USP – São Carlos**
**Março de 2016**

*Dedicated to two great women:*
*My mother, who started me on this journey;*
*and my wife, who kept me on it until the end.*

So we see how we stand then like children at the edge of the ocean of information and we're putting our feet in and wondering, you know, "Could we swim in that? What would it be like to be wet in that? What would it be like to go into that new medium?"

— Terence McKenna, *Final Earthbound Interview*, 1998

# Thesis Summary

Visual analytics tools play an important role in the scenario of ‘big data’ solutions, combining data analysis and interactive visualization techniques in effective ways to support the incremental exploration of large data collections from a wide range of domains. One particular challenge for visual analytics is the analysis of multidimensional datasets, which consist of many observations, each being described by a large number of dimensions, or attributes. Finding and understanding data-related patterns present in such spaces, such as trends, correlations, groups of related observations, and outliers, is hard.

Dimensionality reduction methods, or projections, can be used to construct low (two or three) dimensional representations of high-dimensional datasets. The resulting representation can then be used as a ‘proxy’ for the visual interpretation of the high-dimensional space to efficiently and effectively support the above-mentioned data analysis tasks. Projections have important advantages over other visualization techniques for multidimensional data, such as visual scalability, high degree of robustness to noise and low computational complexity. However, a major obstacle to the effective practical usage of projections relates to their difficult interpretation.

Two main types of interpretation challenges for projections are studied in this thesis. First, while projection techniques aim to preserve the so-called *structure* of the original dataset in the final produced layout, and effectively achieve the ‘proxy’ effect mentioned earlier, they may introduce a certain amount of errors that influence the interpretation of their results. However, it is hard to convey to users where such errors occur in the projection, how large they are, and which specific data-interpretation aspects they affect. Secondly, interpreting the visual patterns that appear in the projection space is far from trivial, beyond the projections’ ability to show groups of similar observations. In particular, it is hard to explain these patterns in terms of the meaning of the original data dimensions.

In this thesis we focus on the design and development of novel visual explanatory techniques to address the two interpretation challenges of multidimensional projections outlined above. We propose several methods to quantify, classify, and visually represent several types of projection errors, and how their explicit depiction helps interpreting data patterns. Next we show how projections can be visually explained in terms of the high-dimensional data attributes, both in a global and a local way. Our proposals are designed to be easily added, and used with, any projection technique, and in any application context using such techniques. Their added value is demonstrated by presenting several exploration scenarios involving various types of multidimensional datasets, ranging from measurements, scientific simulations, software quality metrics, software system structure, and networks.

**Keywords:**
Computer graphics, Visualization, Visual analytics, Multidimensional data

# Resumo

Ferramentas de análise visual desempenham um papel importante no cenário de soluções para grandes volumes de dados (‘big data’), combinando análise de dados e técnicas interativas de visualização de forma eficaz para apoiar a exploração incremental de coleções de dados em diversos domínios. Um desafio importante em análise visual é a exploração de conjuntos de dados multidimensionais, que consistem em muitas observações, sendo cada uma descrita por um grande número de dimensões, ou atributos. Encontrar e compreender os padrões presentes em tais espaços, tais como tendências, correlações, grupos de observações relacionadas e valores extremos, é difícil.

Técnicas de redução de dimensionalidade – ou projeções – são utilizadas para construir, a partir de conjuntos de dados multidimensionais, representações de duas ou três dimensões que podem então ser utilizadas com substitutas do espaço original para sua interpretação visual, apoiando de forma eficiente as tarefas de análise de dados acima mencionadas. Projeções apresentam vantagens importantes sobre outras técnicas de visualização para dados multidimensionais, tais como escalabilidade visual, resistência a ruídos e baixa complexidade computacional. No entanto, um grande obstáculo para o uso prático de projeções vem da sua difícil interpretação.

Dois principais tipos de desafios de interpretação de projeções são estudados nesta tese. Em primeiro lugar, mesmo que as técnicas de projeção tenham como objetivo preservar, na representação final, a *estrutura* do conjunto de dados original, elas podem introduzir uma certa quantidade de erros que influenciam a interpretação dos seus resultados. No entanto, é difícil transmitir aos usuários onde tais erros ocorrem na projeção, quão severos eles são e que aspectos específicos da interpretação dos dados eles afetam. Em segundo lugar, interpretar os padrões visuais que aparecem em uma projeção, além da percepção de grupos de observações semelhantes, está longe de ser trivial. Em particular, é difícil explicar tais padrões em termos do significado das dimensões dos dados originais.

O trabalho desenvolvido nesta tese concentra-se no projeto e desenvolvimento de novas técnicas visuais explicativas para lidar com os dois desafios de interpretação de projeções multidimensionais descritos acima. São propostos alguns métodos para quantificar, classificar e representar visualmente diversos tipos de erros de projeção, e é descrito como essas representações explícitas ajudam na interpretação dos padrões dos dados. Além disso, também são propostas técnicas visuais para explicar projeções em termos dos atributos dos dados multidimensionais, tanto de forma global quanto local. As propostas apresentadas foram concebidas para serem facilmente incorporadas e usadas com qualquer técnica de projeção e em qualquer contexto de aplicação. As contribuições são demonstradas pela apresentação de vários cenários de exploração, envolvendo vários tipos de conjuntos de dados multidimensionais, desde medições e simulações científicas até métricas de qualidade de software, estruturas de sistema de software e redes.

**Palavras-chave:**
Computação gráfica, Visualização, Análise visual, Dados multidimensionais

# Samenvatting

Visuele analyse spelt een sleutelrol in de zogenaamde 'big data' oplossingen door data-analyse met interactieve visualisatietechnieken te combineren om de stapsgewijze exploratie van grote gegevensverzamelingen te ondersteunen. Multidimensionale datasets zijn een aparte uitdaging voor visuele analyse. Deze bevatten veel datapunten, elk dat beschreven wordt door veel dimensies of attributen. Omdat men dergelijke hoog-dimensionale datasets zich niet makkelijk kan voorstellen, het vinden van trends, correlaties, groepen van gerelateerde observaties, en uitschieters is hard.

Dimensionaliteitsreductiemethoden, of projecties, kunnen gebruikt worden om laag (twee of drie) dimensionale representaties van hoog-dimensionale data te bouwen. Deze representaties dienen als hulpmiddelen voor de visuele interpretatie van de hoog-dimensionale ruimte. Projecties hebben een aantal voordelen zoals visuele schaalbaarheid, hoge robuustheid tegen ruis, afstandspreservatie, en een lage rekencomplexiteit. Integendeel is hun interpretatie een grote praktische uitdaging.

Dit proefschrift benadert twee interpretatie uitdagingen. Eerst, alhoewel projecties streven naar het bewaren van de zogenaamde datastructuur, creëren zij altijd een bepaalde mate van fouten, die hun interpretatie beïnvloeden. Het is hard om de gebruikers de locatie, grootte, en invloed (op de data interpretatie) van dergelijke fouten weer te geven. Ten tweede, de interpretatie van de visuele patronen die in projecties voorkomen is verre van triviaal – in het bijzonder het is hard om dergelijke patronen uit te leggen in termen van de oorspronkelijke data dimensies.

Dit proefschrift presenteert het ontwerp en ontwikkeling van nieuwe visuele exploratietechnieken die de twee bovengenoemde interpretatie uitdagingen van multidimensionale projecties verhelpen. We presenteren een aantal methodes voor de kwantificatie, classificatie, en visuele weergaven van verschillende types projectiefouten, en laten zien hoe hun weergave de interpretatie van datapatronen verhelpt. We laten vervolgens zien hoe projecties uitgelegd kunnen worden via hoog-dimensionale attributen op een lokale en ook globale wijze. Onze oplossingen kunnen makkelijk toegevoegd worden aan elke projectietechniek en gebruikt worden in elk toepassingscontext waar dergelijke technieken voorkomen. De toegevoegde waarde van onze oplossingen wordt afgebeeld door verschillende scenario's voor de exploratie van multidimensionale datasets afkomstig uit metingen, wetenschappelijke simulaties, software kwaliteitsmetrieken, softwaresysteemstructuur, en netwerken.

**Trefwoorden:**
Computer graphiek, Visualisatie, Visuele analyse, Multidimensionale gegevens

# Contents

# List of Figures

# List of Tables

Chapter 1

# Introduction

The last decade has witnessed an explosion in the prominence of *data* as a central artifact of scientific and technological development. Multiple facets characterize this unprecedented development. The advent of increasingly sophisticated, accurate, high-throughput and cheap sensing devices and technologies make us, as a society, able to collect amounts of data from a huge range of sources and up to extents that were unthinkable years ago. The increase of computing power and data storage, and decrease of corresponding prices, has made it possible to analyse and explore large amounts of data of virtually any type. These analyses and explorations support a very diverse set of activities, such as validation and verification of models and theories, optimization of algorithms and tools, and obtaining new insights on phenomena described by the underlying data, which can lead to new theories, models, and ultimately socially relevant technological developments.

A critical component of the analysis of this universe – known under the generic name of 'big data' – is the availability of efficient and effective analysis *tools*. These encompass a wide variety of technologies, developed in traditionally separate research domains, such as data mining, knowledge engineering, statistics, databases, business intelligence, mathematical analysis, computer graphics, human-computer interaction, and data visualization. As the problems to be studied become increasingly complex, the size and complexity of datasets describing them increase too. As such, tools and techniques that combine data mining and analysis and interactive exploration techniques are of increasing interest and added value. Such tools are created and used in a new field of expertise called *visual analytics*.

Visual analytics has emerged from the classical fields of scientific and information visualization as a separate discipline focused on the development of theories, methods, techniques and tools for analytical reasoning facilitated by interactive visual interfaces [209]. The field has known a rapid development in the last decade since its advent, which was related to applications in homeland security and business intelligence [37]. Currently, visual analytics tools and techniques are actively being developed and used in areas as diverse as medical science, physics, chemistry, biology, material sciences, geographical information systems, e-governance, and (surely not surprisingly) computer science and information technology [102]. One of the main reasons for the success of visual analytics is its proposal to support incremental sensemaking, in terms of discovering unknown facts in large data collections and next building and refining hypotheses related to the underlying phenomena captured by the data. This is made effective, easy to learn, and generically applicable to a wide range of audiences and domains by using interactive data visualization techniques that present and support the exploration of data in intuitive ways [125].

## 1.1 Multidimensional Data: Importance and Challenges

A particular place in the described context is occupied by *multidimensional data*. Such data typically consists of a set of observations, each being characterized by a number of dimensions, also called attributes. A multidimensional dataset can be thought of as a sampling of a high-dimensional space, whereby the observations play the role of sample points. Multidimensional datasets include, among many other possible examples, population studies (where observations are persons and dimensions are attributes of a person, such as age, gender, profession, salary, job description, or education) [23]; medical studies (where observations are diseases and dimensions are diagnostic and treatment-related data) [133, 67, 9]; software repositories (where observations are files under version control and dimensions include file attributes such as type, size, authors, bug reports, or modification requests) [200, 201].

Multidimensional data pose particularly hard challenges to visual analytics. First, humans find it hard to visually imagine spaces having more than three (or possibly four) dimensions. As such, understanding datasets having hundreds or possibly thousands of dimensions becomes very challenging. Related to this, it is difficult to create depictions, or visualizations, of such spaces. This, in turn, leads to the difficulty of being able to spot patterns, trends, and outliers – which are tasks at the core of the added-value of data visualization. Secondly, multidimensional data may have a mix of attributes of different types and ranges. For instance, our population study example mentioned earlier features attributes which are quantitative (age and salary), categorical (gender, education), and text (job description). Different attribute types require different visual encodings, which makes the design of an effective visualization harder than in the case where all attributes are of the same type. Additionally, reasoning about relative similarities and differences of such distinct attribute types is difficult.

The challenge of multidimensional data has been recognized since long in visual analytics. To address this, several visualization methods have been researched. These range from classical two-dimensional scatterplots to scatterplot matrices, parallel coordinates and space-filling curves. In this family of techniques, *Dimensionality Reduction* (DR) methods, also sometimes known as *projections*, occupy a particular position. Given any set of observations or points, having several dimensions each, and a suitable distance or similarity function defined for them, projections typically construct a two- or three-dimensional scatterplot, under the main constraint of keeping inter-point relationships in the projection space proportional with inter-point relationships in the original high-dimensional space. This allows analysts to employ the scatterplot as a ‘proxy’ for the interpretation of the high-dimensional space in tasks that involve comparing points, such as finding groups or clusters of highly similar points, finding outliers, and detecting trends and correlations.

Projections have a number of important advantages as compared to other multidimensional visualization techniques. First, they scale visually reasonably well in terms of both number of observations and, more importantly, number of dimensions. Secondly, recent projection methods offer a high degree of robustness to noise, distance-preservation accuracy, computational complexity, and ease of use. Thirdly, projection methods exist for datasets having quantitative, categorical, and mixed quantitative-and-categorical attributes [149, 1, 24, 172]. Fourthly, projection plots require typically little to no parameter definitions or choices on the part of the user. This is in contrast to other multidimensional visualization techniques where one has to explicitly decide how to arrange the dimension axes (parallel coordinates, scatterplot matrices). To summarize, we can basically use projections as black boxes to create two-dimensional scatterplots easily, quickly, and efficiently. There are still, however, important challenges related to their usability and ease of interpretation, as described next.

**Projection errors:** *Most projections introduce a certain amount of error that influences the interpretation of their results ("I cannot trust what I am seeing.")*

All projection techniques aim to preserve the *structure* of the high-dimensional dataset. At a low level, this structure is typically quantified in terms of distances between point pairs or, alternatively, in terms of nearest-neighbors of points. Groups of data points exhibiting different local structures can then be seen to form larger-scale data patterns. By preserving this data structure, projections effectively aim to achieve the 'proxy' effect mentioned earlier, *i.e.*, the fact that one can use the resulting low-dimensional scatterplot to reason about high-dimensional similarities or neighbors. Early projection methods, such as Principal Component Analysis (PCA), essentially ignore the variation of all dimensions by describing the data in terms of two components with the highest variance. As such, they introduce considerable errors when projecting any dataset that is not essentially in a hyperplane embedded into a high-dimensional space. More recent non-linear projection methods considerably increase the projection accuracy, quantified in terms of distance and/or neighborhood preservation, by essentially making different decisions on how to project the data for each separated local neighborhood of points [141, 95, 138]. However, even such projections cannot achieve a perfect neighborhood and/or distance preservation, not even for two-dimensional manifold surfaces embedded in high-dimensional spaces – consider, for example, the projection of points spread over the surface of a sphere into a plane.

Given the above, there is often a loss of information that stems from the inability of the DR methods to fully achieve their goal of keeping the original structure of the data intact. Where this information loss occurs and its severity is, however, usually not obvious to the user. Typical projection-quality metrics, such as aggregated stress [139], only provide a global assessment of how accurate a projection is. Such metrics are useful when comparing several different projection techniques, in order to see which one gets *on average* the lowest projection errors. We note that, in this context, projection error metrics are further specialized in metrics that measure distance-preservation errors and metrics that measure neighborhood-preservation errors, in line with the two refinements of the definition of data structure introduced above. However, when users face the concrete task of analysing a *given* projection to make decisions, global aggregate error metrics are less useful. Indeed, consider for example that a neighborhood in the projection might apparently represent very similar points while, in high-dimensional space, those points are not neighbors; or a compact cluster in the projection might actually represent the union of two separate groups of similar points from the original space. Unless explicitly detected and depicted, such local projection errors can easily mislead users when interpreting a projection, thereby severely diminishing the advocated role of the projection as an effective analysis proxy for the high-dimensional dataset.

Apart from showing projection errors, users should also be given ways to *correct* these where and when needed. In operational terms, this can be done by changing the various parameters of a given projection technique, or replacing that technique by a different one. However, to be able to make an informed decision on this, users need to see how parameter variation affects projection errors – or, in other words, one needs effective ways to explore the parameter space of projection techniques. To our knowledge, there is little research that addresses this goal.

**Absence of dimension encoding:** *There is no obvious way of interpreting projected points in relation to the original attributes of the dataset ("I do not know what I am seeing.")*

All mainstream multidimensional visualization techniques, with the exception of projections, explicitly encode (subsets of) the high-dimensional attributes in the resulting visualization. For instance, parallel coordinates have axes that explicitly and unequivocally allow the user to locate the value of any attribute on any data element. When a user sees, for instance, a cluster of edges at a certain region of an axis in such a plot, the interpretation is clear: Those specific data elements (edges) share similar values for that attribute (axis), and the values that are shared are immediately available for inspection by looking at the axis legends. Scatterplot matrices also allow the explicit analysis of any pairs of attributes. After the user locates a desired attribute pair, the data interpretation follows the usage of a standard scatterplot. The trends present in the data regarding the selected attribute-pair can be analysed and, when detected, can be immediately traced back to the original values of the two attributes, located on either one of the axes of the scatterplot. Explicit attribute encoding is also present in scientific visualization. To give just one example, a hedgehog plot shows a three-dimensional vector field (which can be seen as a three-variate dataset) by explicitly encoding the direction and magnitude of the vector attributes by the orientation and length of arrow glyphs [160].

In contrast to the above, standard multidimensional projections only depict the *similarity* of data points, but not their *attribute* values. In a projection, a point's absolute position has no meaning in terms of its original attributes – that is, we cannot infer the values of that point's attributes just by looking at the point itself. If we include in the analysis the neighbors of the point, we may infer various facts about the similarity of points. This tells us *that* the points are similar (and, possibly, how similar these points are), but not *why* the points are similar (that is, due to which attributes and/or attribute values). In other words, the position of a point in a multidimensional projection does not reflect the sole attributes of that point, but the similarity of these attributes with those of neighboring points in high-dimensional space. This is a fundamentally different approach to data encoding than the one used by techniques such as parallel coordinates, scatterplots, scatterplot matrices, or table lenses. The difficulty of interpreting projections is aggravated in cases where we do not see a clear separation between different clusters in the projected image. Overall, it is quite hard for users to locate a region of interest in a projection and comprehend why the technique decided to place those points there rather than somewhere else.

This problem of interpreting, or explaining, projections is well recognized, and a number of solutions have been developed to address it. The simplest solution is to color the points based on the value of a user-selected attribute. While this shows the values of that attribute for all points, and thus allows seeing trends, outliers, and even the individual attribute values for separate points, we need to know beforehand which attribute we want to examine. Examining all attribute values in sequence is not an option, as there may be hundreds of these, and a user's visual memory is clearly limited. Biplot axes are also a well-known tool in statistics [71]. These resemble classical Cartesian-plot axes, and indicate the directions and range of variation of all attributes in a projection. Biplot axes are particularly useful for detecting strongly correlated dimensions. However, classical biplot axes are known only in the context of linear projections, thereby excluding the more recent, high-quality, non-linear projection techniques, *e.g.* [141, 95, 138]. Also, biplot axes do not easily explain the reasons of formation of clusters. Axis legends annotate the two screen axes of a classical linear projection by the amount of variation of each of the original dimensions along these axes, and thereby aim to make the interpretation of a projection scatterplot (more) similar to that of a well-known 2D Cartesian plot [24, 23]. However, such legends do not address structures in the projection that are not axis-aligned. Separately, none of the above-mentioned explanatory techniques has been used for projection scatterplots having more than two dimensions.

## 1.2 Research Questions

From our analysis outlined above, we conclude that projections are efficient and effective instruments for visually exploring high-dimensional datasets, but their practical value is reduced by the absence of explanatory techniques. We focus on two classes of such explanatory techniques, that address (a) the explanation of the quality of projections in terms of fine-grained insight on the projection errors; and (b) the explanation of the meaning of projections in terms of the original high-dimensional attributes.

As such, we can formulate our general research question below:

*How can we increase the added value of multidimensional projections of high-dimensional datasets with explanatory techniques that enable a wide range of users to interpret the information present in a projection in more effective ways?*

We can next refine this research question into two sub-questions, based on our earlier analysis of projection interpretation challenges:

1. How to explain the errors introduced in a projection by the underlying dimensionality reduction technique, so that we see how and where these errors influence our interpretation of the patterns present in the projection?

2. How to explain patterns present in a projection, such as trends, groups, outliers, and correlations in terms of the original high-dimensional attributes?

We observe that these two sub-questions are related, but complementary. Answering sub-question 1 serves the goal of explaining which *elements* present in the projection are trustworthy (and thus should be considered for further exploration), and which not (and thus should be discarded or investigated in other ways). Answering sub-question 2 serves the goal of telling the users how to construct an interpretation of projection regions based on the original data attributes. Additionally, the techniques we will next propose to address our research sub-questions work jointly to help explaining and interpreting a projection, and are complementary – first, separating projection elements that are worth interpreting further from those that carry artifacts of the projection; next, interpreting the elements selected in the previous step. The projection interpretation pipeline afforded by these combined techniques is therefore compatible with the well-known information visualization 'mantra' of Shneiderman [168]: 'Overview first, zoom and filter, then details-on-demand'.

## 1.3 Structure of This Thesis

Below we provide a brief overview of the focus of each following chapter in this thesis.

Chapter 2 presents an overview of visualization methods that address the exploration of multidimensional datasets. As our domain of interest is information visualization, we naturally focus herein on information visualization and visual analytics methods that deal with abstract, non-spatial, data. Also, we focus on the visualization of high-dimensional datasets having hundreds of dimensions. Separately, this chapter provides a survey of multidimensional projection techniques as well as techniques that aim to explain such projections and quantify and present projection errors.

Chapter 3 presents our work towards visualizing and explaining distance-preservation errors that appear in multidimensional projections, as introduced in Sec. 1.1. To this end, we further refine the global notion of distance-preservation errors into several sub-cases, such as false neighbors and missing neighbors, and propose ways to measure and visually explore such errors. We further generalize these error metrics to the coarser-scale of groups of close points in a projection, which typically represent related observations. Separately, we show how we can use the proposed error metrics to explain individual projections, compare different projection techniques, and also explore the parameter space of projection techniques to find optimal parameters in terms of error minimization.

Chapter 4 shows how we adapt our distance-preservation error metrics and visualization techniques introduced in Chapter 3 to quantify and explore neighborhood-preservation errors in multidimensional projections. Similarly to the work presented in Chapter 3, we refine the notion of neighborhood preservation to propose several concrete metrics to quantify this aspect in a projection, and next propose several visual encodings to depict these metrics. We next show how visualizing neighborhood-preservation errors is of practical added value in terms of detecting high-error patterns in a projection that should be interpreted with care when trying to reach global conclusions about the similarity of the projected observations.

Chapter 5 addresses our sub-question related to projection explanation. We focus here specifically on three-dimensional projections that generate, as output, a 3D scatterplot of observations. First, we show how the exploration of 3D projections may achieve, in certain circumstances, a more faithful rendering of high-dimensional structures such as clusters of similar observations than 2D projections, and thereby build a case for the added value of 3D projections. We next identify important sources of difficulty and/or potential errors in terms of 3D projection interpretation, and detail how these relate to the interpretation errors described for two-dimensional projections. We next adapt and extend existing techniques for explaining 2D projections, such as biplot axes and axis legends to the specific context of 3D projections and their interpretation challenges. Finally, we show how our techniques can be used to explain outliers, correlations, and trends in 3D projections in terms of the original high-dimensional variables.

Chapter 6 focuses on the visual explanation of 2D projections. For these, we propose a number of different explanatory techniques than those introduced for 3D projections in Chapter 5, by exploiting the relatively simpler structure of 2D scatterplots. To this end, we first propose several metrics that aim to explain, on a local basis and by using the original high-dimensional attributes, why points in a 2D projection are placed close to each other. Next, we adapt and extend the visual encodings proposed in Chapter 3 to depict our local explanations. While both the techniques introduced in this chapter and Chapter 5 explain projections in terms of the original high-dimensional attributes, our techniques for 2D projections achieve a high local level-of-detail explanation, while the techniques proposed for 3D projections work at a more global level.

Chapter 7 turns our focus to the use of multidimensional projections for a specific concrete application – the study of multivariate networks. We show that we can directly use unmodified dimensionality reduction methods to address the problem of visually presenting a multivariate network in terms of its connectivity, its attributes, or both, solely by adapting its data representation according to the desired goals and feeding it as input to a DR method. We compare the results of this with more traditional force-based algorithms and show how they differ and which features of the network are more or less apparent in each scenario. Finally, we show that, by following this approach, it is then possible to apply the same visual analysis methods introduced in previous chapters to evaluate and interpret this new type of dataset, thus reinforcing the claim of adaptability, applicability and usefulness of the proposed explanatory techniques.

Chapter 8 reflects back on our research questions and the proposed visualization methods for multidimensional projections introduced in the body of this thesis. We summarize our findings in terms of types of projection interpretation challenges *vs* strong points and limitations of the proposed explanatory methods. This chapter also concludes this thesis by summarizing our main findings and also identifying future potential exploration directions for explaining projections, and outline how such methods could improve the ease of use and effectiveness of projections in practical data visualization applications.

# Chapter 2

# Related Work

In this chapter, we present a survey of methods and techniques related to the interactive visual exploration of multidimensional datasets. Within this survey, we dedicate particular attention to multidimensional projection techniques, which offer several important advantages in terms of genericity, visual scalability, and supporting tasks such as detection of outliers and groups of similar observations. In line with our main research questions outlined in Chapter 1, we also discuss techniques for the assessment of projection quality and projection errors, as well as techniques that help explaining projections. The aim of this chapter is to provide a technical background on techniques related to our own work described in the next chapter, and also reflect on advantages and limitations of these techniques in relation to our key research questions outlined in Chapter 1. In turn, these set the stage for the explanation of our proposed techniques for explaining and interpreting multidimensional projections.

## 2.1 Multidimensional Data

Multidimensional data can be described as a representation that associates to each data point $\mathbf{x}_i$, $1 \leqslant i \leqslant M$, also called an *observation*, a set of *attribute values* $x_i^j$, $1 \leqslant j \leqslant N$. The set of all values $x_i^j$ for a given value $j$, over all $M$ observations, is called an attribute, dimension, or feature, and is denoted by $\mathbf{x}^j$. The number $N > 1$ of attributes of a data point is called the dimensionality of the data. Typical multidimensional datasets allow $N$ to take large values. For example, in image classification, a set of images $\mathbf{x}_i$ can be described by a multidimensional feature space, consisting of hundreds of attribute values $x_i^j$ extracted per observation (image) [190, 47]. Similarly, in text analysis, a corpus of documents $\mathbf{x}_i$ is analysed by extracting the frequency of terms (keywords) per document; these generate $N$ attribute values per document, that is, the frequencies of the $N$ considered terms. The use of a large dictionary, containing possibly thousands of terms, can easily generate datasets having thousands of dimensions per observation [13].

Attribute values are associated with types and ranges. In statistical data analysis and information visualization, one distinguishes between several types of attributes, based on the properties of the underlying attribute domain. While not unique and unanimously accepted, the classification presented in the following paragraphs is frequently encountered [125, 181, 73].

***Quantitative***, or continuous, attributes are defined over sets that admit the operations of addition, multiplication with a scalar (real-numbered) value, and ordering of values. Given the above, quantitative attributes are most usually defined over subsets of $\mathbb{R}$, as the real axis meets all above properties. In visualization terms, quantitative attributes give rise to scalar, vector, and tensor attributes [181]. These are sets of attributes that, when taken together, describe the variation of a higher-dimensional quantity. For terminology simplicity and consistency, we will next consider that an attribute $\mathbf{x}^j$ is always one-dimensional (has a single value per observation). Of course, different attributes $\mathbf{x}^j$ can be semantically grouped together to assign a higher-level meaning to the resulting set $\{\mathbf{x}^j\}$, *e.g.* when grouping three scalar attributes to create a three-dimensional vector attribute.

A key property of quantitative attributes is that they allow *interpolation* of values. Indeed, considering that we have two observations $\mathbf{x}_i$ and $\mathbf{x}_j$, then we can create interpolated values between $x_i^k$ and $x_j^k$, for any attribute $k$, as $\alpha_i x_i^k + \alpha_j x_j^k$, where $\alpha \in \mathbb{R}^+$ represent the weights, or interpolation function values, of the two observations $\mathbf{x}_i$ and $\mathbf{x}_j$, measured at the location of the interpolated value. When, moreover, observations are defined over a metric space, which offers us the notion of distance between points, interpolation can generate values that smoothly vary between the attribute values $x_i^k$ at any point located between the observations $\mathbf{x}_i$. Interpolation is a key property for supporting operations such as data resampling, smoothing, filtering, and reconstruction, which in turn support the scalable visualization of large and complex datasets.

***Integral***, or discrete, attributes are defined over sets that admit addition, subtraction, ordering and multiplication by an integral scalar. Such sets are typically ranges of $\mathbb{Z}$ or $\mathbb{N}$. Examples of integral attributes are counts of items, such as number of lines of code measured per software component [107]. Integral attributes typically do not admit interpolation, since that would require multiplication of data values with real numbers, which produces non-integral values.

***Ordinal*** attributes are defined over sets that allow the ordering of values, *i.e.*, support the computation of the relations $<$, $>$, and $=$. Ordinal attributes are usually employed to express relative rankings, *e.g.* measuring the likelihood that one buys a product over {*definitely not, possibly, neutral, probably, definitely yes*}, or ranking the satisfaction of customers of a service over {*unacceptable, bad, poor, neutral, good, very good, excellent*}. Ordinal attributes should not be confused with integral attributes. Indeed, while both admit ordering, it is generally not meaningful to talk about the distance between two ordinal attributes in the same sense we talk about the difference of two integral attributes.

***Categorical***, or nominal, attributes are defined over any set, as they support a single operation – comparing two values to determine if they are equal or different. Categorical attributes are usually employed to encode the notion of category, type, or kind. Examples hereof are, as the name also indicates, types of items, such as gender (male or female) or transportation means (car, plane, train, subway, bicycle). In contrast to ordinal attributes, categorical attributes do not admit a natural 'ranking' of their values – all categories are equally important.

***Text*** attributes are defined over the set of all possible phrases, or productions, that can be generated following a given grammar and given dictionary. Examples hereof are text written in natural language, or source code written following the syntax of a programming language. While text attributes can be seen as categorical (we can easily say when two text fragments are identical) or ordinal (we can lexicographically order strings), they typically come with higher-level semantics. Indeed, not all text fragments in a text dataset are usually considered to have the same importance, and two lexically different text fragments may capture precisely the same semantics; this makes text different from categorical attributes. Separately, it is hard to imagine how to define an ordering of text fragments that reflects any useful text semantics; this makes text different from ordinal attributes.

***Relational*** attributes are defined over sets of observations, and express the fact that two or more such observations jointly participate in a relation that encodes some useful application semantics. Such attributes give rise to datasets known as trees, graphs, and networks. Relational attributes are different from all other attribute types discussed earlier since they need at least two observations to define an attribute value (whereas we associate quantitative, integral, ordinal, categorical, and text attribute-values with a single observation). As such, relational attributes spawn separate families of analysis and visualization techniques, such as graph drawing [44] and graph visualization [202, 78].

At a higher organization level, attribute values for different observations can be grouped, or put in relationship with each other, giving birth to the concept of a *dataset*. Two main types of grouping of observations are described below: *structured* and *unstructured*.

***Structured*** observations correspond to situations where these are ordered in a specific way to represent the sampling of a given domain $D$. The most known instances hereof are the so-called grids. In grids, observations $\mathbf{x}_i$ are organized along a system of so-called structured coordinates that represent the scanning or reading order in which we have to traverse the domain $D$ to yield increasing values of the observation index $i$. The simplest, and most prevalent, examples of grids are 1D, 2D, and 3D uniform grids, which are represented by 1D, 2D, and respectively 3D arrays of observations. These usually represent the sampling of corresponding 1D, 2D, and 3D spatial domains $D$. In this case, the resulting dataset is also called a *field*, and its visual exploration falls under the scope of scientific visualization [73, 181]. However, a grid is not always associated to an underlying spatial domain – consider, for example, the 1D grids formed by ordered sequences of rows (observations) in a table, indexed by row ID; and the 2D grids formed by observations stored in the cells of a matrix.

When structured observations are present, one has to (naturally) use the structure information in the design of suitable visualizations to explore the data. Indeed, to stay with our earlier examples: If a table's rows are to be read in chronological order, and this order is given by the row ID, then a visualization such as a table lens [151] should not sort rows differently. A matrix visualization should reflect the positions of the matrix elements, as this position encodes specific semantics of the elements. Similarly, a field visualization should not re-arrange the samples (observations) in any order, since the positions of observations encode essential spatial semantics.

***Unstructured*** observations correspond to cases where these are not stored in any particular order. As such, the emerging dataset essentially consists of a set (rather than an ordered sequence) of tuples, each representing an observation. This includes all cases of multidimensional data where observations are not taken in any particular order, or where the sampling order has no particular semantics. Examples hereof are population studies, where an observation encodes all data (attributes) available over a person, and there is no semantics associated to which person was recorded (sampled) first or next. Unstructured observations are harder to visualize than structured observations, as they are non-spatial by excellence – we have absolutely no information in the dataset on how to arrange, or *embed* these observations into the (2D or 3D) visualization space. In contrast, such information is available for structured observations, in terms of relative order or even precise spatial location.

Unstructured observations of high dimensionality (tens to hundreds of attributes) and potentially having all attribute types are one of the key focus areas of multidimensional visualization, and the main focus of this thesis. In this context, several types of visualization methods have been proposed. We next briefly review these methods, in order to better place in context the added-value and challenges of multidimensional projections – our area of interest.

## 2.2 Multidimensional Visualization Tasks and Methods

Multidimensional visualization methods can be characterized by the types of *tasks* they support. In contrast to more generic task characterizations, such data-specific task characterizations are of added value as they provide a more explicit mapping between concrete tasks and specific (visualization) tools that address these tasks, and thereby facilitate coupling tasks with tools when desigining visualization applications [21, 161]. However, while characterizations and taxonomies of multidimensional data visualization *techniques* exist, a comprehensive and universally accepted characterization of analysis *tasks* for multidmensional data is still not available. This makes it hard to reason about individual data-analysis tasks, and even harder in the case of sequences of such tasks (also called workflows) that address multidimensional data [176, 88, 94].

In a recent paper, Brehmer *et al.* characterized the visual exploration of multidimensional data as a combination of five types of tasks: (1) naming synthesized dimensions, (2) mapping a synthesized dimension to original dimensions, (3) verifying clusters, (4) naming clusters, and (5) matching clusters and classes [22]. This categorization is very interesting in our context of using multidimensional projections to visually analyse such data. Indeed, tasks (1) and (2) directly map to our first research sub-question (Sec. 1.2) that refers to explaining patterns present in a projection in terms of the original data dimensions; task (3) directly maps to our second research question (Sec. 1.2) that refers to quantifying and displaying projection errors that may influence the interpretation of patterns present in a projection; and tasks (4) and (5) are essentially a refinement of our first research sub-question, specialized for cluster patterns in a projection. We shall further address tasks (1) and (2) by our explanatory visualizations of 3D projections in Chapter 5; task (3) by our explanatory visualizations for projection errors in terms of distances (Chapter 3) and neighborhoods (Chapter 4); and tasks (4) and (5) by our local attribute-based explanatory visualizations in Chapter 6.

A further useful categorization of multidimensional visualization methods splits these into observation-centric and attribute-centric methods, as follows:

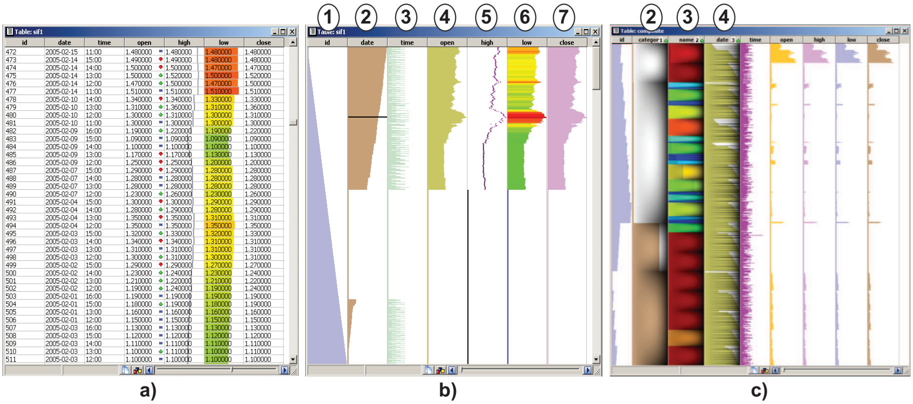
- *Observation-centric* methods focus on explicitly depicting observations as easy-to-recognize, compact, items in the resulting visualization. Such methods are best for tasks that revolve around the identity of observations, such as finding groups of similar observations or outlier observations. Conversely, such methods are less effective for reasoning about attributes, *e.g.* finding direct or inverse correlations of attributes or finding how attribute values evolve over the range given by all observations;
- *Attribute-centric* methods focus on explicitly depicting the variation of all attributes in the resulting visualization. Such methods are best for tasks related to the analysis of attributes, such as finding direct or inverse correlations or finding maximum or minimum values. Conversely, such methods are less effective for reasoning about observations, *e.g.* assessing the similarity of two or more observations.

As we shall see below, most (if not all) multidimensional visualization methods propose different trade-offs between observation-centric and attribute-centric visualizations, but none offers optimal ease in reasoning about *both* observations and attributes.

### 2.2.1 Table Lenses

Tables are probably the oldest, and best-known, technique for visualizing multidimensional data. Consider a multidimensional dataset $X = \{\mathbf{x}_i\}$ where each observation is a tuple $\mathbf{x}_i = (x_i^1, \ldots, x_i^N)$ of $N$ attribute values. A table visualization essentially draws each observation $\mathbf{x}_i$ as a horizontal row in which the attribute values $x_i^j$ are explicitly shown by text in cells. This layout, hence, creates columns for each separate attribute $\mathbf{x}^j$. When the number of observations $M$ or attributes $N$ becomes too large to fit the table's visual size, scrolling is used to go to the desired location. Besides rendering attribute values as plain text, these can be also encoded into bars scaled and colored to reflect the attribute value. Figure 2.1a shows this for a data table where each row encodes a stock transaction, having as attributes the name of the traded stock, category of the stock, trade price, trade moment (day and intraday time), and trade volume [180]. This color-and-size coding enables one to pre-attentively spot *e.g.* maxima or other outlier values when quickly scrolling through a large table.

To address scalability in the number of observations, the table lens technique proposes next to reduce the size of a row to one pixel size [151]. This effectively renders each observation as a one-pixel-thick horizontal line, which allows thousands of such observations to fit on a modern computer screen without scrolling. In this case, using scaled and/or color-coded bars to convey attribute values is mandatory (Fig. 2.1b). The table lens supports additional tasks besides making the visualization scalable, such as detecting trends and correlations. For instance, in Fig. 2.1b, we can easily see that variable 1 is linearly increasing top-to-bottom, while variables 2, 4, 5, 6, and 7 show a strongly correlated pattern. Additionally, different color encodings can be used for the different variables, to *e.g.* highlight specific aspects. An enhancement atop of the classical table lens is the possibility to sort and group observations by attribute values [180]. Figure 2.1c shows an example: Here, the observations (stock transactions) are grouped first by category, next by stock name, and finally by trading date. Same-value blocks of



| id | date | time | open | high | low | close |
|---|---|---|---|---|---|---|
| 472 | 2005-02-15 | 11:00 | 1.480000 = | 1.480000 | 1.480000 | 1.480000 |
| 473 | 2005-02-14 | 15:00 | 1.490000 ▼ | 1.490000 | 1.480000 | 1.480000 |
| 474 | 2005-02-14 | 14:00 | 1.500000 ▼ | 1.500000 | 1.470000 | 1.470000 |
| 475 | 2005-02-14 | 13:00 | 1.500000 ▲ | 1.520000 | 1.500000 | 1.520000 |
| 476 | 2005-02-14 | 12:00 | 1.470000 ▲ | 1.500000 | 1.470000 | 1.500000 |
| 477 | 2005-02-14 | 11:00 | 1.510000 = | 1.510000 | 1.510000 | 1.510000 |
| 478 | 2005-02-10 | 14:00 | 1.340000 ▼ | 1.340000 | 1.330000 | 1.330000 |
| 479 | 2005-02-10 | 13:00 | 1.310000 ▲ | 1.360000 | 1.310000 | 1.360000 |
| 480 | 2005-02-10 | 12:00 | 1.300000 ▲ | 1.310000 | 1.300000 | 1.310000 |
| 481 | 2005-02-10 | 11:00 | 1.300000 = | 1.300000 | 1.300000 | 1.300000 |
| 482 | 2005-02-09 | 16:00 | 1.190000 ▲ | 1.220000 | 1.190000 | 1.220000 |
| 483 | 2005-02-09 | 15:00 | 1.090000 = | 1.090000 | 1.090000 | 1.090000 |
| 484 | 2005-02-09 | 14:00 | 1.100000 = | 1.100000 | 1.100000 | 1.100000 |
| 485 | 2005-02-09 | 13:00 | 1.170000 ▼ | 1.170000 | 1.130000 | 1.130000 |
| 486 | 2005-02-09 | 12:00 | 1.250000 ▼ | 1.250000 | 1.200000 | 1.200000 |
| 487 | 2005-02-07 | 15:00 | 1.290000 ▼ | 1.290000 | 1.280000 | 1.280000 |
| 488 | 2005-02-07 | 14:00 | 1.280000 = | 1.280000 | 1.280000 | 1.280000 |
| 489 | 2005-02-07 | 13:00 | 1.280000 = | 1.280000 | 1.280000 | 1.280000 |
| 490 | 2005-02-07 | 12:00 | 1.230000 ▲ | 1.260000 | 1.230000 | 1.260000 |
| 491 | 2005-02-04 | 15:00 | 1.300000 ▼ | 1.300000 | 1.290000 | 1.290000 |
| 492 | 2005-02-04 | 14:00 | 1.280000 ▲ | 1.290000 | 1.280000 | 1.290000 |
| 493 | 2005-02-04 | 13:00 | 1.350000 ▼ | 1.350000 | 1.310000 | 1.310000 |
| 494 | 2005-02-04 | 12:00 | 1.350000 = | 1.350000 | 1.350000 | 1.350000 |
| 495 | 2005-02-03 | 15:00 | 1.320000 ▲ | 1.330000 | 1.320000 | 1.330000 |
| 496 | 2005-02-03 | 14:00 | 1.340000 ▼ | 1.340000 | 1.310000 | 1.310000 |
| 497 | 2005-02-03 | 13:00 | 1.310000 = | 1.310000 | 1.310000 | 1.310000 |
| 498 | 2005-02-03 | 12:00 | 1.300000 ▲ | 1.310000 | 1.300000 | 1.310000 |
| 499 | 2005-02-02 | 15:00 | 1.290000 ▼ | 1.290000 | 1.270000 | 1.270000 |
| 500 | 2005-02-02 | 14:00 | 1.230000 ▲ | 1.240000 | 1.230000 | 1.240000 |
| 501 | 2005-02-02 | 13:00 | 1.210000 ▲ | 1.220000 | 1.210000 | 1.220000 |
| 502 | 2005-02-02 | 12:00 | 1.190000 ▲ | 1.240000 | 1.190000 | 1.240000 |
| 503 | 2005-02-01 | 16:00 | 1.190000 = | 1.190000 | 1.190000 | 1.190000 |
| 504 | 2005-02-01 | 15:00 | 1.180000 ▲ | 1.190000 | 1.180000 | 1.190000 |
| 505 | 2005-02-01 | 13:00 | 1.160000 = | 1.160000 | 1.160000 | 1.160000 |
| 506 | 2005-02-01 | 12:00 | 1.150000 = | 1.150000 | 1.150000 | 1.150000 |
| 507 | 2005-02-03 | 16:00 | 1.130000 = | 1.130000 | 1.130000 | 1.130000 |
| 508 | 2005-02-03 | 15:00 | 1.120000 = | 1.120000 | 1.120000 | 1.120000 |
| 509 | 2005-02-03 | 14:00 | 1.110000 = | 1.110000 | 1.110000 | 1.110000 |
| 510 | 2005-02-03 | 13:00 | 1.100000 ▲ | 1.110000 | 1.100000 | 1.110000 |
| 511 | 2005-02-03 | 12:00 | 1.100000 = | 1.100000 | 1.100000 | 1.100000 |

Figure 2.1: Table lens visualization of stock transactions created with the TableVision tool [180]. The images shown the zoomed-in table (a), zoomed-out aggregated table (b), and zoomed-out table with rows sorted and grouped by stock category, name, and trading date (c).

contiguous rows are emphasized in the visualization by using shaded cushions [196]. The resulting hierarchy visualization is known also under the name of icicle plots [106]. This type of visualization allows one to focus the analysis on specific attribute value-ranges, and also to see how the attribute values distribute over the entire observation set.

Table lenses are quite effective in showing the individual evolution of each attribute $\mathbf{x}^j$ and, up to a certain extent, in highlighting correlations between different attributes. They also scale very well to hundreds of thousands of observations, by using aggregation techniques that average contiguous rows in the final table [180]. However, obtaining an insightful and, ultimately, useful table lens visualization strongly depends on finding a good order of the rows that places observations of interest close to each other. Doing this may not always be possible, since we cannot sort the table in the same time on multiple attributes. As such, tasks such as finding groups of strongly similar observations (with respect to all their attributes) are not directly supported by table lenses. Separately, table lenses do not readily scale to datasets having hundreds of attributes.

### 2.2.2 Small multiples

Besides tables, small multiples are another well-known technique for visualizing multidimensional data [187]. The key idea is simple: Consider a multidimensional dataset $X = \{\mathbf{x}_i\}$ where each observation is a tuple $\mathbf{x}_i = (x_i^1, \ldots, x_i^N)$ of $N$ attribute values. Assume next we have a visualization method for all values, over all observations, of a given attribute $1 \leqslant j \leqslant N$. Denote by $V_j$ the visual representation of this attribute $\mathbf{x}^j$. A small multiple visualization is, then, a spatial organization of all visualizations $V_j$, $1 \leqslant j \leqslant N$, also called *multiples*, that allows one to compare and correlate values of different attributes. To enable this, the visualizations $V_j$ should not only have a similar design, but also use similar visual encodings, *e.g.* color, line thickness, scaling, shading, and texture. However, it is not mandatory that *all* these encodings be identical. As long as the task of comparing different visualizations $V_j, V_{k \neq j}$ is supported, a few visual encodings can be used to mark aspects that are different for different attributes $j$.

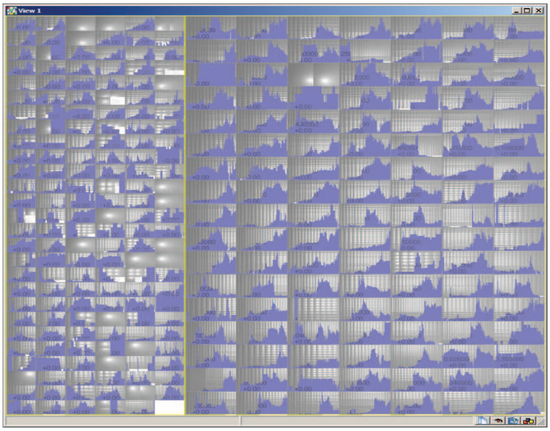


Figure 2.2: Small multiple visualization of stock transactions created with the TableVision tool [180].

Classical examples of small multiple visualizations include timelines and bar charts organized in grids, which are ubiquitous in business intelligence and infographics in general. Figure 2.2 shows a slightly more advanced small multiple visualization along these lines [180]. The input dataset consists of the same type of data table as discussed earlier in Sec. 2.2.1. The table lens visualization discussed in Sec. 2.2.1 regarded these data as a multidimensional dataset where an observation is a stock *transaction*. At a higher level, we can look at the same data as being a multidimensional dataset where each *stock* is an observation, and its attributes are its category, name, and *evolution* of trade price and volume over a given time period. Following this model, the data are first organized into a hierarchy, by grouping observations by category, next by stock, and finally by trading day, as outlined in Sec. 2.2.1. The resulting structure is visualized using a cushion treemap [196]. This already shows the appearance of two top-level cushions or blocks of observations, which represent each one of the two stock categories present in the input data. Next, a classical timeline graph of the per-day average transaction price is drawn atop of the treemap cushion of each stock. As the stock cushions are laid out in lexicographic order (top-to-bottom and left-to-right, in alphabetical order of the stock name) and using equal sizes when constructing the second treemap level, this effectively creates a small-multiple layout where we can compare the evolution of daily stock prices (attributes) across different stocks (observations). Separately, since stocks (observations) are lexicographically ordered on name, we can create a mental map of where a given stock is located based on its name.

Small multiples are effective in allowing one to flexibly organize a number of visualizations $V_j$ so as to capture a wide range of aspects, *e.g.* attribute values, attribute names, or domain-specific conventions. This favors creating visualizations that have a 'natural' reading order. Note that table lenses can be seen as an (extreme) particular case of a small multiple visualization, where each visualization $V_j$ depicts a different table column or attribute $\mathbf{x}^j$. A limitation of small multiples is that they cannot cope with more than a few tens of visualizations $V_j$ – therefore the qualification 'small'. This creates problems when one maps each attribute of a high-dimensional dataset to a separate multiple. Separately, the ordering of multiples will favor comparing those that are closely placed in the visualization, and make it hard(er) to compare those that are far away. A more subtle effect is that, if multiples are ordered in a 2D grid-like fashion, then this order should be taken into account in conjunction with the natural reading order of a grid-like 2D infographic, which is largely lexicographic [84]. Finally, just as for table lenses, small multiples are largely an attribute-centric, rather than an observation-centric, technique – they allow comparing values of attributes, but do not (typically) show observations explicitly in the visualization. As such, tasks where observations are central, such as finding groups of strongly similar observations or finding outlier observations, are not optimally supported by small multiples.

### 2.2.3 Scatterplot methods

Scatterplot techniques are, as a visualization means for multidimensional data, on par with tables and small multiples in terms of popularity. Classical scatterplots are simple and easy to explain: Given a multidimensional dataset $X = \{\mathbf{x}_i\}$ with observations $\mathbf{x}_i = (x_i^1, \ldots, x_i^N)$, a two-dimensional scatterplot $S_{jk}$ plots a 2D point $\mathbf{p}_i = (x_i^j, x_i^k)$ for each observation $\mathbf{x}_i$, using as 2D Cartesian coordinates the values $x_i^j$ and $x_i^k$ of $\mathbf{x}_i$ along two user-selected attributes, or dimensions, $\mathbf{x}^j$ and $\mathbf{x}^k$. The result is a familiar 2D point cloud whose shape and spread allows one to interpret the data along the two plotted attributes in various ways: The plot extent, or size, conveys an idea of the range of the attributes; the plot's spread conveys the correlation strength of the two attributes; the plot's closeness to a line conveys the strength of linear correlation (or inverse correlation, depending on slope) of the attributes; and isolated points in the plot indicate outlier observations.

The main strength of 2D scatterplots lies in their simplicity and ease of construction. Most importantly, the values of the two plotted attributes $\mathbf{x}^j$ and $\mathbf{x}^k$ are directly visible along the plot's $x$ and $y$ axes respectively. Constructing such plots is straightforward for quantitative and integral variables that admit a direct mapping to the real-valued plot axes. For categorical variables, scatterplots can be constructed by using multiple correspondence analysis techniques [1, 71], which we will discuss in more details in Sec. 2.3. Clutter can, however, occur in 2D scatterplots, when multiple observations project atop, or very closely to, each other [49]. In such cases, a classical way to reduce clutter and convey more insight is to display a 2D density map of the points $\mathbf{p}_i$, computed *e.g.* by kernel density estimation, and visualized by a color coding [195]. While this does not make all observations distinguishable, it conveys a clear idea of how many observations one can see in every region of the scatterplot, and thus makes densely-populated scatterplot regions, which are arguably more interesting, more salient.

The main limitation of 2D scatterplots is related to the fact that they can only show two attributes at the same time. A third attribute can be added atop a classical

2D scatterplot, by color-coding the points $\mathbf{p}_i$ by their value. However, this interferes with densely-populated point regions and creates clutter that cannot be solved by the density map solution outlined earlier. Alternatively, three attributes can be visualized by constructing a 3D scatterplot in analogous ways to 2D scatterplots. This results in 3D point clouds which can be then explored interactively from multiple viewpoints to find correlations, trends, and outliers. However, interpreting 3D scatterplots is generally found harder than interpreting 2D scatterplots, due to effects such as occlusion, depth ambiguity, the difficulty of assessing distances between arbitrary 3D points, and the need for the user to choose a suitable viewpoint [156]. To ease the interactive exploration of 3D scatterplots, several techniques have been proposed, such as controlled animation when changing viewpoints [158, 86] and illumination models that highlight the local point-cloud density distribution (curve, surface, and volumetric) differently to ease perception of dense scatterplots [157]. However, even with these refinements, visually exploring 3D scatterplots still remains challenging.

To increase the scalability, in terms of dimensions being shown, of scatterplots, different techniques have been proposed. Scatterplot matrices (or SPLOMs) create a small-multiple-type visualization consisting of $N \times N$ cells organized in a symmetric matrix [12]. Each matrix cell $(j, k)$ shows the 2D scatterplot $S_{jk}$ of variables $j$ and $k$. To facilitate the correlation of various attributes of the same observation in all the matrix cells or multiples, the *linked views* technique is used – upon brushing or selecting an observation, or set of observations, in any cell, the same observations are highlighted and/or selected in all other cells of the SPLOM. While this offers some support of correlating the different 2D scatterplots, SPLOMs still remain an attribute-centric technique. In particular, finding groups of strongly related (similar) observations requires an iterative process of interactively selecting close points in one or several multiples and checking whether the selected points are close in all other plots. Separately, SPLOMs do not scale well with the number of attributes, as they essentially require $\frac{N^2}{2}$ cells to show an $N$-dimensional dataset.

Several refinements have been proposed to address the limitations of SPLOMs. One such approach is to exploit interactivity to make the navigation of the user in the high-dimensional space easier and/or more effective for specific tasks. In this sense, the 'rolling the dice' technique allows users to continuously transition between two SPLOM cells $S_{jk}$ and $S_{lm}$, by creating an animation that linearly interpolates the positions of the plotted points $\mathbf{p}_i$ between the two plots [50]. Specific interaction tools are provided to facilitate the navigation between cells that share one attribute axis, *e.g.* $j = l$ or $k = m$. An additional advantage of this technique is that it limits the required visual space – in the limit, one can use only a single 2D scatterplot view to navigate the entire data space. The idea of user-controlled animated interaction between two scatterplot-like views was extended next by Hurter *et al.* [87, 86], in terms of allowing the user to stop the animation at any desired stage, and also interactively control the playback speed and direction. This allows one to locate interesting patterns in the animated scatterplot that may be visible only at specific animation stages, and also select such patterns to explore further. Overall, such interactive techniques move a part of the data space to be explored from a spatial encoding (like in SPLOMs) to a temporal encoding (animation). While this effectively increases scalability in terms of dimensions, it also requires additional learning and temporal-memory efforts from the users.

A separate approach to enhance SPLOMs is proposed by a group of techniques collectively known as *scagnostics* [188]. In essence, these techniques pre-analyse the original high-dimensional data or, alternatively, the 2D scatterplots that a SPLOM would create, and try to detect a small number of views on the data that best depict a number of predefined patterns of interest. Next, these views are displayed using *e.g.* a classical SPLOM. The two advantages of this approach are reducing the number of views shown to the user to the most 'interesting' ones; and, since views were selected based on some relevance metric that aims to recover patterns, annotating the views to explain which types of patterns they depict. Different types of metrics aim to locate different types of patterns, such as outliers, shapes, trends, density, and coherence of scatterplots [207]; or relevance and coherence of 2D scatterplots [110]. While scagnostic techniques can clearly reduce the number of views required to depict high-dimensional data, they are implicitly limited in the power of the underlying analysis techniques to detect which such views are 'interesting'. For relatively simple patterns such as outliers and correlations, this is quite easy to do. However, patterns that have more complex shapes or, even more challengingly, which require multiple variables to be described, are hard and/or computationally costly to detect. On a higher level, one can also state that such approaches are not effective when one needs to explore a dataset without any *a priori* idea or expectation on the patterns to be found – in such cases, we want to 'discover the unknown' rather than searching for the known (patterns), so different approaches are needed.

### 2.2.4 Parallel Coordinate Plots (PCPs)

Parallel coordinate plots (PCPs) [90] aim to make observations more visible while preserving the possibility to compare and analyse the data from an attribute-centric perspective. For a multidimensional dataset $X = \{\mathbf{x}_i\}$ with observations $\mathbf{x}_i = (x_i^1, \ldots, x_i^N)$, a PCP constructs $N$ parallel axes, one for each variable $\mathbf{x}^j$. Next, the values of the variables $x_i^j$ are plotted on each corresponding axis. Finally, the plotted points that correspond to the same observation $\mathbf{x}_i$ are connected by a polyline. The resulting plot contains, thus $M$ polylines that cross the $N$ axes at locations indicating their attribute values.

Figure 2.3 illustrates the tasks supported by PCPs by showing two visualizations created with the PCP technique by using the *parvis* visualization tool [108]. Groups of similar observations (in all dimensions) appear as thin bands of nearly parallel polylines. These can be further emphasized by computing a density map by drawing polylines using alpha blending. Flat distributions of attribute values show up as axis intersection points spread uniformly along a given axis. In contrast, attribute-value distributions that have localized peaks show as 'bundles' of polylines that intersect the respective attribute axis at the given attribute value. Outliers appear as isolated polylines that do not visually group to a band. Correlated values of attributes plotted on adjacent axes appear as polyline fragments consisting of nearly parallel lines. In contrast, inversely correlated values (for the same types of axes) appear as polyline fragments that form an X-like pattern between the axes. Histograms of attribute values can be overlaid on the axes to give a more quantitative view of the attribute-value distributions than when looking at alpha-blended polylines. Finally, interaction such as brushing and selection supports tasks such as finding all observations that have a given range of attribute values.

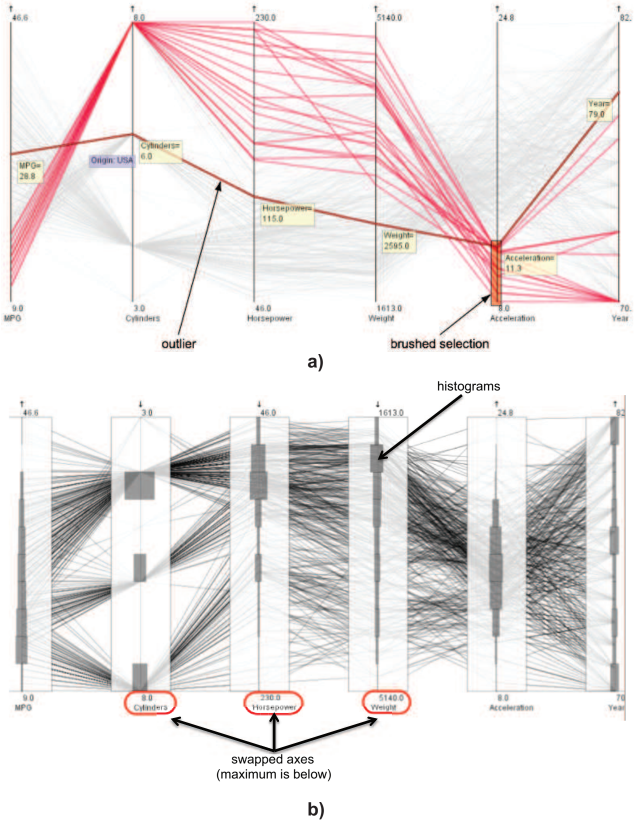


Figure 2.3: Parallel coordinate plot using the *parvis* toolkit for a 6-dimensional dataset (images from [181]).

Several enhancements to the basic PCP design have been proposed. Axes can be permuted and swapped (in terms of direction) to limit the clutter created by intersecting polylines, and also to bring axes that one wants to examine in conjunction close to each other. Axes can also be aligned in a radial, rather than parallel, pattern, leading to the so-called radial plots [81], star plots [30] and star coordinates [99, 100] (which can be also seen as a variant of scatterplots). This flexibility of arranging axes is taken a step further by FLINAPlots, which essentially allow users to draw and arrange axes interactively as they desire, so as to position axes relatively to each other in ways that best emphasize the desired patterns.

PCPs and their variations are effective in striking a good balance between the visual predominance of observations and attributes, allowing users to pursue both observation-centric and attribute-centric exploration strategies. However, they also come with several limitations. First, the clutter created by numerous crossing polylines can lead to displays where one can hardly see both observations and attribute variations. This can be explained in terms of the amount of *ink* being used to draw a given dataset of $M$ $N$-dimensional observations and the amount of *overlaps* present in the respective drawing: Table lenses have zero overlaps, and use an amount of ink that is proportional to the product $N \cdot M$. Small multiples have also zero overlap, while using a similar amount of ink. SPLOMs have zero overlap and use an amount of ink proportional to $M\frac{N^2}{2}$. In contrast, PCPs have significant overlap in general, and use an amount of ink which, while being also of the order of $M \cdot N$, is higher than for table lenses and, for low values of $N$, also higher than for SPLOMs, since one needs to leave a significant amount of space between consecutive axes to draw the respective polyline segments with limited clutter. A second limitation of PCPs regards their linear structure, or axis arrangement. While this structure favors comparing adjacent axes, it makes it quite hard to compare axes that are situated far apart from each other in the plot. Permuting axes can bring different pairs of axes close to each other, but cannot, for instance, generate a design where three or four axes are *equally* close to each other. Separately, one needs to know how to arrange axes beforehand in order to highlight specific patterns – and if such an arrangement is unknown, otherwise interesting patterns may be missed. Finally, just as the other multidimensional visualization techniques discussed earlier, PCPs are quite limited in the maximal number of dimensions they can visualize simultaneously.

## 2.3 Dimensionality Reduction

As we have outlined in the previous sections, visualizing multidimensional data is challenging, especially when the number of dimensions is high. In particular, considering the characterization of such visualization methods into observation-centric and attribute-centric introduced at the beginning of Sec. 2.2, we see that table lenses, SPLOMs, and small multiples are all largely attribute-centric methods. Indeed, SPLOMs and small multiples do not offer a single and compact visual encoding (depiction) of an observation, but plot it as a set of spatially disjoint elements in the visualization. Table lenses do plot an observation as a spatially-contiguous row of cells (encoded as colored bars); however, rows for different observations cannot be easily compared, as they are typically ordered in the table by the value of a single attribute; also, a row itself is far from being compact, thus hard to be recognized, and reasoned about, as being an observation. From all techniques reviewed above, PCPs come closest to being observation-centric, as they explicitly represent observations as polylines. However, similar to table lenses, the spatial extent (in the resulting visualization) of an observation is quite large, so it is relatively hard to reason about the data from an observation-centric perspective.

Dimensionality reduction (DR) methods, also called *projections*[1], propose a diametri-

[1] We note that the term *projection* is often used in the literature to refer to both a dimensionality reduction method (a function) and the results of applying this method to some high-dimensional dataset (the result of a function). We will, accordingly, use the term projection with the same semantics, and let the reader disambiguate its two senses from the context.

cally opposed view on a multidimensional dataset. Here, observations are clearly and easily identifiable as points in a typical 2D or 3D scatterplot. More formally, given a multidimensional dataset $D^n = \{\mathbf{p}_i \in \mathbb{R}^n\}_{1 \leqslant i \leqslant N}$ of $N$ $n$-dimensional points, a dimensionality Reduction (DR) method can be seen as a mapping

$$f : \mathbb{R}^n \times P \to \mathbb{R}^m \tag{2.1}$$

that maps each point $\mathbf{p}_i \in D^n$ to a point $\mathbf{q}_i \in D^m$. Here, $n$ is typically large (tens up to thousands of dimensions), and $m$ is typically 2 or 3, corresponding to the generation of typical 2D or 3D scatterplots respectively, which can be directly drawn for visual inspection. $P$ denotes the *parameter space* of $f$, *i.e.* the various settings that control the projection algorithm $f$ and affect its output.

The goal of any DR method is to keep various aspects of the so-called *structure* of the multidimensional data as similar as possible in $\mathbb{R}^n$ and $\mathbb{R}^m$. By data structure, we mean here aspects that are relevant for the application at hand, such as the distance between observations or the $k$ nearest neighbors of an observation (for a user-specified value of $k$). If such aspects are kept (nearly) identical in the original high-dimensional data $D^n$ and in the projection space $D^m$, or in other words if the projection *preserves* such aspects, then one can use the projection space as a 'proxy' to reason about the invisible high-dimensional space. For instance, if we are interested to find groups of very similar observations, and our projection preserves distances (similarities) between observations, then we can visually locate such groups in $D^m$ and thereby directly find observations that are similar in $D^n$. A crucial aspect that enables this task is the fact that projections preserve, by construction, the fact that an observation is a point in *both* $D^m$ and $D^n$. In other words, projections are by excellence observation-centric methods.

### 2.3.1 Multidimensional Scaling (MDS) methods

A particular case of DR methods forms a subset of the wider family of data-analysis methods known under the name Multidimensional Scaling (MDS) [120, 40]. The key aspect of these methods is that they do not need access to the original high-dimensional coordinates of the observations in $D^n$ (the original attribute values of each instance of the dataset). Rather, all they need is a real-valued distance matrix $A^n = (d^n(\mathbf{p}_i, \mathbf{p}_j))_{ij}$, where $d^n$ is a distance metric over $D^n$. These methods aim to compute a distance matrix $A^m = (d^m(\mathbf{q}_i, \mathbf{q}_j))_{ij}$, where $d^m$ is typically Euclidean distance in the projection space $D^m$, and $\mathbf{q}_i$ is the projection of observation $\mathbf{p}_i$, so that $A^m_{ij}$ is proportional to, or a scaled version of, $A^n_{ij}$. Hence, the name multidimensional scaling. If this is achieved, we say that the DR method under consideration preserves distances, which is, as outlined earlier, a desirable feature for projections.

Given the distance matrix $A^n$, these methods compute the DR function $f$ by minimizing an aggregate badness-of-fit measure [17, 40] such as the commonly used *normalized stress* function

$$\sigma = \frac{\sum_{1 \leqslant i \leqslant N, 1 \leqslant j \leqslant N} (d^n(\mathbf{p}_i, \mathbf{p}_j) - d^m(\mathbf{q}_i, \mathbf{q}_j))^2}{\sum_{1 \leqslant i \leqslant N, 1 \leqslant j \leqslant N} (d^n(\mathbf{p}_i, \mathbf{p}_j))^2} \tag{2.2}$$

If a distance matrix is not available, it can be straightforwardly computed from the observations $D^n$ themselves. However, computing (and storing) distances this way creates additional costs ($O(N^2)$ for $N$ points). The PLMP algorithm avoids this by using distances only for a small set of representative points and using additional information present in the $nD$ coordinates to arrange the remaining points around these representatives [141].

DR methods using an MDS approach can be classified by the techniques used to compute $f$ [141]. *Spectral decomposition* techniques project points on lower-dimensionality subspaces computed by considering specific eigenvectors of the point-wise distance matrix [186]. LLE [152] and ISOMAP [184, 170] use efficient numerical methods tailored to solve sparse eigenproblems to speed up such computations. Landmarks MDS [169] and Pivot MDS [20] achieve further speed-ups by using classical MDS on a subset of representative points and projecting remaining points by local interpolation. Fastmap achieves linear complexity in the input observation count but has a worse stress minimization [55]. MetricMap improves the computational speed of Fastmap by trying to perform the entire projection in a single step [203]. Fastmap, MetricMap, and Landmark MDS have been shown to be instances of the same minimization framework [147].

A different approach to compute MDS-based projections is to construct a graph whose nodes are the original observations and edges connect (ideally) all observation pairs. Nodes are assigned random positions in $\mathbb{R}^m$. Edges are next assigned weights which are inversely proportional to the distances $d^m(\mathbf{p}_i, \mathbf{p}_j)$ corresponding to the observation pair $(\mathbf{p}_i, \mathbf{p}_j)$, and these weights are regarded as elastic spring energies. Next a spring embedder algorithm is run to diminish the total energy of the graph, much like in classical graph-drawing algorithms [44]. Heuristics and optimizations are proposed to increase the convergence speed of this relaxation and to obtain a graph whose mode positions essentially minimize the stress $\sigma$ (Eqn. 2.2) [208].

### 2.3.2 Coordinate-based projections

A different class of projection methods directly uses the coordinates of the observations $\mathbf{p}_i \in D^n$ rather than their distance matrix. As outlined earlier in Sec. 2.3.1, this has the advantage, upon MDS methods, of storing only $O(N \cdot n)$ values as opposed to $O(N^2)$ values. For datasets having many more observations than dimensions, this produces considerable savings. However, the disadvantage of these methods, which we next call coordinate-based projections, is that they are less general than MDS methods, as they need to have access to the original observations in $D^n$, which can be of various attribute types (Sec. 1.1).

One of the earliest coordinate-based projection methods is based on the Karhunen-Loèwe transform [60]. For this, the covariance matrix of the observations' coordinates in $D^n$ is computed. Next, the $m$ eigenvectors thereof corresponding to the $m$ largest eigenvalues are found, and the observations in $D^n$ are projected on the hyperplane defined by these vectors. This yields the point set $D^m$ which captures most of the variance of the input data [97]. This technique has, however, the main disadvantage that it assumes that the observations are well captured by a $m$-dimensional hyperplane. As such, in cases where the observations are still embedded on an $m$-dimensional curved surface (manifold), which is however not a plane, the projection will loose a significant amount of the data variance.

Nonlinear optimization methods iteratively search the parameter space $P$ to minimize the stress $\sigma$ [62, 155]. Besides naive gradient descent, multigrid numerical solvers can be used to speed searching [25]. Pekalska *et al.* propose a speed-up that projects a representative subset (by gradient descent) and fits remaining points by local interpolation [144]. Force-based methods are a special class of nonlinear optimization with many uses in graph drawing [48]. Chalmers speeds up such methods by using the representative subset idea outlined earlier [29]. Further speed-ups are achieved by multilevel solvers and GPU techniques [59, 89], and by recursively selecting representatives via a multilevel approach [98]. Tejada *et al.* use a heuristic to embed instances by force-based relaxation [179]. LSP positions the representative subset by a force-based scheme and fits the remaining points by Laplacian smoothing [139]. LAMP also uses a representative subset to locally construct affine projections, and allows users to interactively place these points to optimize the overall projection layout [95].

Apart from computational speed-up, most representative-based methods (but not all – LSP [139] is an exception) have the important advantage of being able to *locally* define the projection. Assuming that, in the high-dimensional space $D^m$, data is locally placed on a quasi-planar manifold, such projections can therefore minimize the stress $\sigma$ (Eqn. 2.2) by taking different decisions for each representative. This allows them to project high-dimensional data that is embedded on curved manifolds much more accurately, in terms of distance preservation, than global methods such as PCA-based techniques [95].

Some techniques, on the other hand, use a mix of both coordinates and distances in order to achieve their results. One recent example is the t-Distributed Stochastic Neighbor Embedding (t-SNE) [192], an improvement over the Stochastic Neighbor Embedding (SNE) [80] technique that introduced the idea of converting the pairwise distances between each pair of points $i$ and $j$ as conditional probabilities, both in the high-dimensional ($p_{i|j}$) and the low-dimensional ($q_{i|j}$) space, then minimizing the divergence between the two in order to find a faithful low-dimensional representation for the data. While SNE computes the probabilities in both spaces using Gaussian distributions, t-SNE uses a heavy-tailed Student t-distribution in the low-dimensional representation in order to make sure that dissimilar points end up far away from each other, avoiding the crowding problem and leaving more space for similar points to form well-defined clusters, while also achieving a cost function that is easier to optimize.

The work presented in this section is a necessarily incomplete review of multidimensional visualization techniques (and implicitly dimensionality reduction techniques), due to space limitations. However, for the purpose of establishing the working context of this thesis, the presented material gives a sufficient overview of the relevant space of methods and techniques. Additional details on specific techniques will be given in the context of our work presented in the next chapters, as necessary.

The discussion in this section shows that projections are highly interesting tools for examining multidimensional datasets. Indeed, they are far more visually scalable than other related techniques such as table lenses, SPLOMs, small multiples, and PCPs.

In the limit, we can say that projections offer the ultimate scalability in *both* number of observations and number of dimensions – an observation is represented by a single $m$-dimensional point. They are also fast to compute, and work largely as black boxes, *i.e.*, require minimal or even no explicit parameter choices from the users. However, projections also have several drawbacks and challenges, which are intimately related to their attempt to reduce a high-dimensional observation-set to a low-dimensional point-set. We next present related work on how projections cope with the two main challenges outlined in our research questions in Sec. 1.2, *i.e.*, understanding the quality of a projection (discussed below), and explaining projections in terms of data attributes (Sec. 2.5).

## 2.4 Challenge 1: Visualizing Projection Quality

As mentioned earlier in Sec. 2.3, projections *attempt* to create a low-dimensional dataset $D^m$ that reflects the structure of the high-dimensional dataset $D^n$. The key word to reflect on here is 'attempt'. Indeed, in many cases, it is very hard, or sometimes even impossible, to create a low-dimensional embedding $D^m$ whose inter-point distances are precisely proportional to their high-dimensional counterparts – consider, again, the example of projecting points located on the surface of a 3D sphere to a 2D plane, a problem well-known from classical cartography. As such, projection errors, or non-zero values of the stress $\sigma$ (Eqn. 2.2), are unavoidable. Such errors should be explicitly acknowledged *and* visually represented, to enable users to reason about how they want to proceed next, *e.g.*, adapt their interpretation of the projection as a 'proxy' for the high-dimensional data, or even discard the projection (and attempt to generate a better one) if errors are too large and/or too frequently occurring.

Given the huge range of projection techniques and heuristics, 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 interpret high-dimensional data by means of a projection [193, 118, 159, 8], and is done at three levels of detail.

At the coarsest level, error metrics like normalized stress [17], correlation coefficients [63], and silhouette coefficients [137] capture the overall projection quality by a single number. This allows globally comparing projection methods [135, 53], but does not show how errors are distributed over the various points in a given projection. In other words, global metrics do not show projection problems for *any* point $i$ *vs all* points $j \neq i$ in the input dataset. For instance, a relatively low aggregated projection error may be an insignificant signal for further interpretation of the projection: If the error is spread uniformly over a projection containing many observations, then we *locally* get a very low distance distortion, which is arguably negligible for further projection interpretation. If, however, the error is concentrated in a single location, then one should refrain from making any judgments on the projection close to and around that location.

On a finer detail level, distance scatterplots show the correlation of distances in $D^n$ *vs* $D^m$ for every point-pair [95]. A similar technique is proposed to measure cluster segregation [164]. Neighborhood-preservation plots (NPPs) show how a projection preserves neighborhoods, for all possible neighborhood sizes [139, 199, 11, 198]. Of these, the trustworthiness-preservation metric [198] 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, NPPs 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 aim to show projection errors for each projected point $\mathbf{q}_i$ in the projection $D^m$, and thus give the most insight in how projection errors are distributed and may affect the interpretation of the resulting visualization. These methods can be classified into methods that focus on distance-preservation errors (Sec. 2.4.1) and methods that focus on neighborhood-preservation errors (Sec. 2.4.2).

### 2.4.1 Distance-preservation errors

Recognizing that DR methods can create distance approximation errors, Van der Maaten *et al.* extend the t-SNE technique [192] to output a set $\{M_i\}$ of 2D projections rather than a single one [193]. All points appear in all projections $M_i$, with potentially different weights and at different locations. This allows better modeling non-metric similarities. Yet, correlating points over the several $M_i$ is done manually by the user, and can be challenging for large datasets and many projections $M_i$.

Several quality metrics for continuous DR techniques are proposed by Aupetit [8]. Point-based stretching and compression metrics measure, for each $\mathbf{p}_i \in D^n$, the aggregated increase, respectively decrease, of the distances of its projection $\mathbf{q}_i \in D^2$ to all other projections $\mathbf{q}_{j \neq i}$ *vs* the distances of $\mathbf{p}_i$ to all other points $\mathbf{p}_{j \neq i}$. Segment stretching and compression measure the variation of distances of close point pairs $(i, j)$ between $\mathbb{R}^n$ and $\mathbb{R}^2$. For a selected $\mathbf{p}_i$, the proximity metric maps distances in $\mathbb{R}^n$ from $\mathbf{p}_i$ to all other points $\mathbf{p}_{j \neq i}$ to the corresponding points $\mathbf{q}_i \in \mathbb{R}^2$ and thereby helps understanding how (and where) the projection may have distorted the structure of the data. These metrics are visualized with piecewise-constant interpolation of the point, respectively segment, data using Voronoi diagrams.

Still using colored Voronoi cells, Lespinats and Aupetit show, at the same time, point stretching and compression by using a 2D color-coded map [111]. The proposed colormap encodes stretching as green, compression as purple, low-error points as white, and points with high stretching and compression as black, respectively. While this color map can show local error types (or the absence thereof), it cannot explicitly show the point-pairs that cause stretching and compression. Besides, as the authors also note, Voronoi cells can lead to visualization bias due to the cells' sizes and shapes being heavily dependent on the $D^2$ point density, and the fact that cells cover the entire $\mathbb{R}^2$ space, even in areas where no projected points exist. A similar remark was made by Broeksema *et al.* in the context of their related usage of Voronoi diagrams to color-code categorical data displayed by projections [24, 23] (see Fig. 2.4 for an example hereof).

To assist the task of navigating projections while also considering distortions, Heulot *et al.* present an interactive semantic lens that filters points projected too closely to a user-selected focus point in $\mathbb{R}^2$ [79]. Such points, also called *false neighbors*, are pushed towards the lens border, so they do not attract the user's attention. Separately, points are colored by the distance in $D^n$ to the focus point, to help users navigate to the so-called *missing neighbors* of the focus point. Instead of Voronoi cells of [8, 111], points are colored using Shepard interpolation, which yields a smoother, and arguably less distracting, image. However, in contrast to [8, 111], this method can only show errors related to a single selected focus point.

### 2.4.2 Neighborhood-preservation errors

In contrast to distance-preservation metrics, *neighborhood-preservation* metrics aim to find how $k$-nearest neighborhoods are affected by the projection. Neighborhood preservation analysis must cover two cases [198]: 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 $D^m$ [159]. 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 [118], undetected. Motta *et al.* propose a neighborhood validation metric that 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* [124]. While related to our set-difference view (Sec. 4.1.3), F-measures are computed by using an extended MST graph [123] 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 defined using $k$-nearest neighbors are, by construction, a function of the number $k$ of considered neighbors. Most such metrics let $k$ be a free parameter, to be set by the user. However, choosing an appropriate value for $k$ is not evident, similarly to what happens in other applications in graphics or data visualization that use similar neighborhoods, such as point cloud reconstruction [32] or 3D shape processing [18]. Indeed, in such graphics applications, $k$ plays the role of a scale parameter, *i.e.*, *monotonically* increasing $k$ will continuously enlarge the scale at which the considered shape is processed, *e.g.* by ignoring or smoothing out small-scale details. Contrary to these applications, however, neighborhood preservation does not necessarily behave in a similar monotonic manner with respect to the parameter $k$: A point may have the same neighbors in both $D^n$ and $D^m$ for low $k$ values; separately, for $k = N$, where $N$ is the total number of points in our dataset, neighborhoods are obviously perfectly preserved. However, for intermediate $k$ values, neighborhood preservation may be lower. Since neighborhood preservation is a non-monotonic function of $k$, the question arises how should the user choose a meaningful $k$ value to assess this desirable property of projections. To our knowledge, this question has not been definitively answered by existing research.

## 2.5 Challenge 2: Explaining Projections

As outlined several times in this chapter, DR techniques are essentially observation-centric: They explicitly show observations (as projected points in $D^m$), but do not show attributes. This creates several interpretation challenges. Consider, for instance, one of the main tasks for which projections are used – finding groups of similar points. It can be argued that, if a projection has low errors (as discussed in Sec. 2.4), then such groups can be easily found by visually spotting clusters of close points separated by the remainder of the projected points by large white space gaps. However, upon having detected such a group, a key question ensues: *Why* are these points similar? Which attributes make them related? Such questions, equally relevant for outliers, need to be addressed. We discuss next several classes of methods related to this goal.

**Interactive approaches** explain projections by showing additional information on-demand on user-selected points or point groups to help one define their meaning. The simplest such technique shows the attribute values of the point under the mouse in a tooltip. By brushing a point-group, one can (hopefully) see which dimensions are most similar and, thus, likely capture the group's meaning. A second simple and popular technique lets users color-code all points in $D^m$ by the value of a single attribute. If this attribute takes very similar values for the points of interest, then it is likely a good explanation for their similarity. Brushing and color-coding are arguably some of the best known and most frequently used techniques in data and information visualization applications, and are present in all major visualization systems. As an example, see the Projection Explorer tool [140]. However, this technique requires one to cycle through all attributes of the input dataset to find the one that best explains similarities. Also, if similarity is explained by several, rather than a single, attribute, this technique is less suited. A different approach is proposed by the ForceSPICE tool, which uses a force-directed spring model to lay out a scatterplot of textual elements [51]. The content similarity of each document can be further inspected and the user can incrementally add annotations over the layout or highlight specific text words. These actions update the spring model to change the layout, to better reflect the user's mental model. Cuadros et al. [41] use a phylogenetic tree algorithm to project documents by placing similar ones in close nodes of the tree. Next, users can execute a topic extraction algorithm that automatically labels selected tree branches to guide exploration. Such approaches explain an projection on several levels of detail, but require user interaction effort to specify *where* to explain the projection.

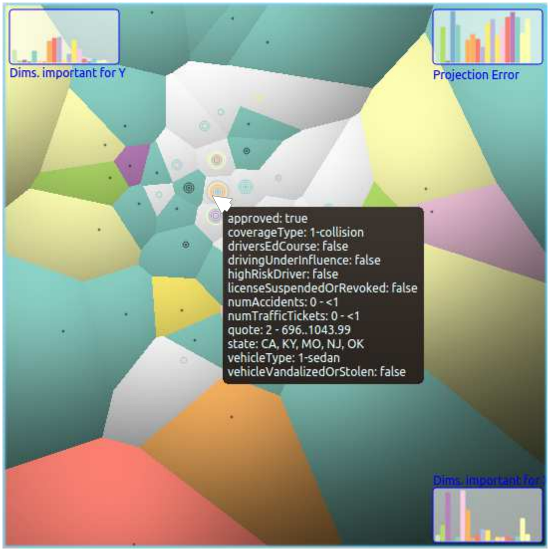
**Clustering** can be used to separate the projection $D^m$ into closely-related point groups. Projecting clusters instead of individual points creates various multi-level visualizations where each projected cluster can be potentially explained by one or a few 'representative elements' drawn atop of it using glyphs. ImageHIVE [175] applies this idea by defining clusters from a collection of images. Using representatives of each cluster, a graph is created based on the $n$D distances between images, which is next drawn in 2D using a graph layout technique. A Voronoi diagram is used to show the representatives' contents. Multi-level maps are also used to visualize documents [130]. The document corpus is projected and clustered by a hierarchical clustering method. Cluster representatives are used to create a Voronoi diagram filled with representative words. Showing representatives, however, does not explain, in terms of attributes or dimensions, why documents are placed together. Kandogan [101] visually annotate clusters occurring in scatterplots based on the attribute trends detected in them. Clusters are computed by an image-based scatterplot density estimation. Important attributes are identified based on their statistical relevance. The ProjCloud technique [143] presents an approach to build word clouds inside general polygons formed by the outline of clusters from projections, while still preserving the semantic relationship among keywords and their connection to their underlying document sources. The clusters can be generated manually – in which case the technique can be considered an *Interactive Approach* – or automatically. These approaches work well when the data and projection can be easily and robustly separated into several clusters, and less well when there is no such clear separation. The work of Joia *et al.* [96] improves on previous clustering techniques by first identifying representative instances of the dataset – with the use of Singular Value Decomposition (SVD) – then clustering all the remaining instances according to their nearest representative. This method provides more sensitivity to data variability and unbalanced data sets (those with clusters of very different sizes). The same mathematical framework used to identify representative instances can also be adapted to identify the most representative attributes of each cluster. Concluding, it is important to notice that clustering-based approaches are inherently sensitive to the type and parameters of the clustering technique used. As such, users can generate multiple potentially different explanations of the same dataset $D^n$, with the ensuing interpretation ambiguities.

**Biplots** (and their variations) assign a meaning to the $m$ dimensions of the $D^m$ projection space in relation to the original $n$ attributes [71, 69]. A biplot generalizes the scatterplot idea of plotting bi-variate observations with respect to two Cartesian coordinate axes, into many variables being observed and viewed simultaneously. Consider the $N \times n$ matrix $\mathbf{D} = (\mathbf{p}_i)_{1 \leqslant i \leqslant N}$. If $\mathbf{D}$ has rank $r$, it can be rewritten by singular value decomposition (SVD) as

$$\mathbf{D} = \mathbf{U}\Delta\mathbf{V}^{\mathrm{T}} \tag{2.3}$$

where $\mathbf{U}$ is a $N \times r$ matrix, $\Delta$ is a $r \times r$ diagonal matrix of eigenvalues $\alpha_1 > \ldots > \alpha_r > 0$, and $\mathbf{V}$ is a $n \times r$ matrix. Here, $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Denoting $\mathbf{F} = \mathbf{U}\Delta$, we have $\mathbf{D} = \mathbf{F}\mathbf{V}^{\mathrm{T}}$. The columns of $\mathbf{V}^{\mathrm{T}}$ define the biplot *axes*. The rows of the left matrix $\mathbf{F}$ define the *projections* of our data points $\mathbf{p}_i$ onto these axes. If $r \leqslant 3$, we can directly visualize the biplot by drawing projections as a point cloud and biplot axes as vector glyphs (oriented straight lines) respectively. If $r > 3$, we can approximate $\mathbf{D}$ by using in Eqn. 2.3 only the first $m < r$ columns of $\mathbf{U}$ and $\mathbf{V}$. Then $\mathbf{F}$ gives the $m$-dimensional projections $\mathbf{q}_i$ of $\mathbf{p}_i$ along the eigenvectors corresponding to the $m$ largest eigenvalues $\alpha_1, \ldots, \alpha_m$ of $\mathbf{D}\mathbf{D}^{\mathrm{T}}$. Using eigenvectors as biplot axes, however, does not convey much insight, as eigenvectors usually do not relate one-to-one to the original variables in $D^n$. A better solution is to construct $n$ biplot axes by projecting, via Eqn. 2.3, the $n$D unit vectors having one for each variable value and zero for all other $n-1$ variables. This overlays our $m$-dimensional scatterplot with $n$ vectors that show the direction of maximal variation of our $n$ variables [71, 1].

A different approach is given by Broeksema *et al.* [24]. Here, a $n$D categorical dataset is projected to $m = 2$ dimensions by SVD. Instead of drawing $n$ biplot axes, the contributions to the screen $x$ and $y$ axes of all original $n$ dimensions are shown. These contributions, also called *loadings* [71, 1], are the projections of the $n$D unit vectors (via Eqn. 2.3) on the two eigenvectors that determine the projection. The $x$ and $y$ axes are annotated with two $n$-element bar-charts or axis legends, where the height of each bar shows the contribution of a given variable to the respective axis (Fig. 2.4). A third bar chart shows the contributions of all $n$ variables to all eigenvectors *not* used to construct the DR projection. This shows the amount of information not captured by the 2D projection. A similar visualization of loadings is shown in [132].



approved: true
coverageType: 1-collision
driversEdCourse: false
drivingUnderInfluence: false
highRiskDriver: false
licenseSuspendedOrRevoked: false
numAccidents: 0 - <1
numTrafficTickets: 0 - <1
quote: 2 - 696..1043.99
state: CA, KY, MO, NJ, OK
vehicleType: 1-sedan
vehicleVandalizedOrStolen: false

Figure 2.4: Axis legends explaining the loadings of a multidimensional projection on the $x$ and $y$ axes of the corresponding 2D embedding space. Visualization created with the Decision Exploration Lab [23].

**Incremental explanation approaches** address both the quality analysis and the explanation of a projection jointly. The early VIBE system allowed users to freely place in 2D space several so-called points of interest (POIs), each representing a sample of the $n$-D space under study [134]. Points in this space represent documents along $n$ dimensions encoding term frequencies. Actual documents are placed in the same 2D space so as to reflect their relative similarities with the given POIs. Conceptually, this can be seen as projecting both documents and POIs (variable values) from $n$-D to 2D. However, this approach requires the user to manually create relevant POIs (samples of the $n$-D space) and also place them suitably in 2D. ForceSPIRE, a document-exploration system, uses a force-based layout to construct a 2D projection of a set of documents represented as $n$-D term vectors [51]. By dragging, pinning, and annotating documents, users can incrementally assign higher-level *semantics* to 2D inter-document distances. The 'dust & magnets' (D&M) technique extends the exploration power of ForceSPIRE and VIBE by allowing users to interactively drag magnets to discover how data points (dust) are attracted towards them in an animated fashion [210]. However, all these techniques require a non-negligible amount of interaction effort from their users, and are as such less appropriate for automated data visualization construction.

**3D projections**, *i.e.*, projections where the target space $D^m$ has $m = 3$ dimensions rather than the more common $m = 2$ two-dimensional, classical scatterplot-like, projections, pose particular interpretation challenges. They typically achieve a better distance and/or neighborhood preservation than their 2D counterparts. This is not surprising, as 3D projections have one extra dimension in which to embed the high-dimensional data variation contained in the input dataset. As such, 3D projections are, technically, an interesting alternative to the more common 2D projections for the task of high-dimensional data exploration. However, 3D projections come with their own challenges, which are discussed next.

A first aspect to consider is whether 3D projections are more suitable to certain tasks than their 2D counterparts. This is still an open subject [178]. Several authors argue that 2D DR plots are better for visualizing text documents [128, 205], and that 2D navigation is easier than its 3D counterpart [205]. For the specific task of cluster separation, Sedlmair *et al.* argue that 2D DR plots are found to be as good as (interactive) 3D DR plots [163]. 2D DR plots were also found better for search tasks [206] and for tasks involving distance assessment and spatial arrangements [54]. On the other hand, Jolliffe argues that 3D projections are needed to "encode a realistic picture of what the data look like" when the intrinsic data dimension is 3 or higher [97]. Dang *et al.* show how 3D glyph stacking can overcome color coding problems in 2D plots [43]. Additional cues such as illumination and depth are proposed in support of using 3D scatterplots [157]. Sanftmann *et al.* argue that high-point densities in scatterplots are better handled by 3D scatterplots [158]. Chan *et al.* argue that 3D projections decrease information loss by allowing better discrimination between data elements [31]. A discussion of contexts where 3D DR projections are preferable to 2D ones is given in [156] (Sec 2.3). Poco *et al.* compared 2D and 3D DR projections using LSP [139] both quantitatively (by stress metrics) and qualitatively (by controlled user studies) [148]. The quantitative comparisons showed a higher accuracy of 3D projections; the user studies showed that, when augmented by suitable interaction tools, 3D projections were superior to 2D projections in terms of both confidence and satisfaction, and argued for the further development of 3D interactive exploration tools.

Assuming that 3D projections are, indeed, desirable from a projection-error minimization and task-suitability perspective, the next question to address is how to interactively explore these. One main issue here is how to choose a suitable viewpoint that highlights specific data aspects or, alternatively, which data aspects are visible from a given viewpoint. This issue has been partially addressed by multiple views, such as three 2D views linked with a 3D scatterplot by interactive selection [146]. While not strictly speaking a technique to explore 3D projections, 'Rolling the dice' comes very close to this goal, as it adds interactivity to improve navigation, 3D animated transitions to explore the visual space, and swapping the scatterplot-matrix axes to show variable correlations and disparities, much like rotating a 3D scatterplot [50]. This idea was extended in [158] by linking a 3D scatterplot with a 3D scatterplot matrix, improving navigation by using three axes and using one or two axes during visual transitions. Claessen *et al.* [34] extend axis movement for scatterplot navigation, to allow users to interactively draw, place, and link axes on a canvas, thereby creating a continuous combination-space of 2D scatterplots, scatterplot matrices, and parallel coordinates. Although the method is very flexible, it can create visualizations with redundant (replicated) axes. Also, similar to [50], while the overall effect is close to what one would obtain by actually rotating a 3D scatterplot, this technique is inherently two-dimensional. Separately, techniques as biplots [71] and axis legends [24] have not been extended to handle 3D projections.

It is important to highlight that 3D multidimensional projections and 3D scatterplots have related, but not identical, understanding challenges. Issues such as occlusion, difficulty of estimating distances between 3D points, the challenge of selecting a good viewpoint, and depth ambiguities, mentioned in Sec. 2.2.3 for 3D scatterplots, are also shared by 3D projections. Consequently, the methods mentioned in the same Sec. 2.2.3 that aid 3D scatterplot exploration are also applicable to 3D projections. However, 3D projections have additional challenges that are not shared by 3D scatterplots. The main such challenge is the lack of (simple) semantics for the dimensions of the embedding 3D space – whereas, for a 3D scatterplot, each such dimension maps a given attribute, a dimension maps a weighted sum of attributes for a 3D projection (for details, see the discussion on loadings in Sec. 2.5). Secondly, the placement of every point in a 3D scatterplot is exact with respect to the mapping of its attribute values to embedding space dimensions. This is not the case for 3D projections that are affected by distance approximation errors (Sec. 2.4).

## 2.6 Multivariate Networks

A multivariate network, also called a multivariate graph, is a graph $G = (V, E = V \times V)$ of vertices, or nodes, $V$, and edges $E$. Such a graph is called multivariate if nodes $n \in V$ and/or edges $e \in E$ have several associated attributes of the types earlier discussed in Sec. 1.1, *i.e.*, quantitative, integral, ordinal, categorical, or text.

Understanding multivariate networks is particularly challenging, as these combine data of two fundamentally different natures: *relations* (which are defined on pairs of nodes) and *scalar attributes* (which are defined on individual nodes and/or relations). In particular, to understand relations, we need suitable ways to depict them. This is the domain of graph visualization or graph drawing [44, 202, 78]. Many algorithms have been proposed to depict the structure of graphs. Many such algorithms are able to cope with specific constraints such as large graphs of millions of nodes [62], reducing edge-crossing clutter to show the main graph structure [82, 61, 85], and using specific drawing styles or conventions to highlight specific graph sub-structures [70]. However, the large majority of graph visualization techniques are severely limited when they have to consider showing several attributes per node and/or edge together with the graph structure [46]. Given our interest in multivariate data visualization, such use-cases are of clear interest. In the following paragraphs, we outline existing methods for visualizing multivariate graphs, with a particular focus on the visualization of social networks, which are a prime example of multivariate graphs.

**Classical node-link techniques** are the most prominent type of visualization of multivariate graphs, and offer precedence, in terms of the visualization layout construction, to the relational information (as opposed to the attribute information). In simple terms, such techniques construct a graph visualization to display the network structure, and next propose various mechanisms to add the visualization of node and/or edge attributes atop of the existing graph layout. For example, Huisman and Duijn describe several prominent tools for the visual exploration of social networks [83]. These include NetDraw, able to visualize large social networks; StoCNET, MultiNet, UCINet and Agna, which perform statistical analysis on social networks; and Blanche and Condor, capable of simulating the network evolution. This survey also points out that only few techniques and tools exist that are able to visually depict attributes of vertices and/or nodes; and that such visualizations are usually limited to classical encodings of a few attributes into shape, color, and size, similar to well-known techniques in general-purpose multivariate graph visualization [46, 7]. A tool offering similar capabilities is Vizster which, atop of classical graph-based visualization of networks and encoding a few attributes into node and edge visual properties, also offers mechanisms to mine for patterns of interest that may be present in the network [75].

The challenge of visualizing multivariate networks can be split into two sub-challenges: the visualization of multiple attributes per node, respectively the visualization of multiple attributes per edge. Several attributes can be shown for a single node (in a node-link metaphor) by encoding them to several visual dimensions (shape, size, texture, size, labels). This approach is relatively scalable, in the sense that one can encode a large number of attributes per node if the node size, in the visualization, is large enough. In the limit, for instance, nodes themselves can become full-fledged visualizations of multivariate data tables [27]. However, this implies a trade off between the number of nodes and the number of attributes per node that a given visualization can accommodate. In the case of edges, the multivariate challenge is far harder: It is much more difficult to design a visual encoding that represents several attributes for an edge in a node-link metaphor. Solutions such as encoding attributes into edge thickness, shading, transparency, and color work reasonably well only for networks having a small number of edges (to be more precise, edge intersections) [46]. As an alternative, more space can be allocated to edges in the visualization, by using a table-lens, edge-centric, visualization [150]. However, this metaphor is less intuitive than the classical node-link depiction of networks. As such, visualizing networks with more than two or three attributes per edge is, to date, an open challenge.

**Adjacency matrices** are an alternative way to node-link layouts to display relational data. Given a graph $G = (V, E = V \times V)$, an adjacency matrix is a symmetric square matrix of $\|V\| \times \|V\|$ elements, where element $(i, j)$ indicates whether node $i$ is connected with node $j$, or whether $(i, j) \in E$. Adjacency matrices have the key advantage that they can display moderately-sized densely-connected graphs with zero occlusion, something that node-link layouts usually cannot do [2]. In turn, this is an interesting feature in case one wants to show multivariate attributes for nodes and/or edges. An example of this approach is MatrixExplorer, which allows users to view a network as a set of nodes and edges in coordination with a matrix representation [76]. The tool also creates a view of connected components, where each component is viewed as a compact rectangle whose size and color reflect the number of vertices contained in that component. NodeTrix represents a network as a mix of adjacency matrices and classical node-link layouts [77]. For this, the input network is separated into several so-called communities (sets of strongly-connected nodes), which are further linked by edges via nodes that belong to different communities. Relationships between community members are represented by adjacency matrices (which are a compact way to depict strongly-connected components in a graph). Relationships between communities are depicted as node-link edges between the respective adjacency matrices representing the communities containing the involved nodes. The visual properties of the vertices and edges can be modified to reflect attributes of the network.

**Multiple linked views** offer a different approach to tackle the problem of visualizaing the high-dimensional space formed by relations and attributes. The basic idea of this approach is simple: Each such view addresses a subset of the input data, created by 'slicing' either in the number of observations (nodes and edges) or in the number of dimensions (node and/or edge attributes). Interactive linking of the views by brushing and/or selection allows one next to correlate various data aspects. In this context, Namata *et al.* presented a tool, called DualNet, capable of generating coordinated representations or linked views [127]. The tool treats the network as a set of sub-networks, each of which can be viewed and manipulated independently in different coordinated views. The tool also allows the modification of visual properties of the vertices to reflect the attributes of the network. Shen *et al.* propose the OntoVis tool, which is designed for the visualization of heterogeneous networks, *i.e.*, networks in which vertices represent more than one type of object [166]. From an ontology graph, containing the different types of objects, users can construct a derived graph by including only nodes whose types are selected in the ontology graph. Next, node sizes and colors can be modulated to reflect attributes such as node centrality and node types. Perer and Shneiderman described a system, called SocialAction, that uses attribute ranking and coordinated (linked) views to help users examine a number of social network analysis metrics [145]. Users can iterate through visualizations of metrics, aggregate networks, find cohesive subgroups, and focus on communities of interest, as well as separate networks by viewing different link types individually, or find patterns across different link types using a matrix overview. Aris and Shneiderman [5] developed the Substrate Designer, a tool for users to specify attributes for grouping nodes into non-overlapping regions, and attributes for placing them within regions. Users can specify the graph drawing algorithm and decide on how to encode additional attributes into visual parameters of nodes and/or edges, thereby facilitating several network analysis tasks. Li and Lin [114] proposed a mechanism for egocentric information abstraction in heterogeneous social networks. They extract a set of features from a given 'ego' node based on linear combinations of its edges, and calculate statistical dependency measures between these features and the ego node. After filtering, they generate a condensed feature graph representation as the abstraction of the given ego node.

**Similarity encoding:** Several social network analysis tools attempt to display nodes according to attributes or similarities. The key idea here is to use the attribute-based similarity of nodes, computed as in the case of multidimensional projections (Sec. 2.3), to create visualizations of multivariate networks. At a high level, the paradigm offered here reverts the approaches described above, where the network (relational) data is central in driving the visualization layout, while the attributes are 'retrofitted' in a subsequent pass on the ensuing visualization. Rather than doing this, the approaches discussed next jointly consider attributes and relations in the creation of the visualization, thereby obtaining a view that reflects both types of attributes equally well.

In this context, one approach is described by Gloor *et al.*, which aims at visualizing social networks as a graph where the positioning of vertices is based on the similarity of the content of the messages exchanged by the actors [68]. Velardi *et al.* also presented an approach to grouping nodes based on the content of the messages exchanged by the actors of the network [197]. The Graphdice tool employs various multidimensional visualizations in support to visual analysis of social network, including one type of projection technique [15]. Smith *et al.* proposed an approach based on attributes to computing a degree of similarity between actors [171]. For each attribute, a network can be generated in which the weight of the edges reflects the degree of similarity calculated for that attribute. Other approaches to representing graphs based on vertex and/or edge attributes are described by Pretorius and Van Wijk [150] and Archambault *et al.* [4]. These approaches define a hierarchical structure based on the attributes to view the network, similar to the grouping of table rows by equal-attribute values presented in the TableVision tool [180]. The resulting hierarchy is next visualized with a variation of one-dimensional treemaps, known also as an icicle plot [106]. Wattenberg *et al.* propose a scatterplot in which the axes are determined by the dimensions of a query summarization [204]. All actors who take the same values for both axes are grouped into a single node, and its size reflects the amount of grouped actors.

However, the realm of creating visualizations that encode both relations and attributes is far from being exhausted. Current approaches, such as the ones outlined above, take different design decisions with respect to how to encode the similarity of attributes *vs* how to encode the relational connectivity patterns; how to weigh the impact of the attribute *vs* relation-based similarity, or connectivity, or nodes; and how to display the final result, which aims to reflect both similarity in attributes and connectivity of nodes. Different options for combining attributes and relations in determining the placement and/or rendering of nodes and edges are clearly possible and needed, and thus worth studying.

## 2.7 Discussion and Conclusions

The review of related work on multidimensional data visualization presented in this chapter offers a challenging picture of the state-of-the-art in this field *vs* desired requirements. Many techniques for visualizing multidimensional data exist, including table lenses, small multiples (or linked views), scatterplot matrices, parallel coordinate plots, and dimensionality-reduction techniques. Analysing the behavior of all these methods, we notice that there is a lack of optimal support for both observation-centric and attribute-centric analysis tasks.

Multidimensional projections have several important advantages over the other techniques reviewed in this context. They are by excellence scalable both in the number of observations and dimensions, are robust and computationally scalable, utilize a very simple visualization metaphor (2D or 3D scatterplots), and can be used with minimal effort from the viewpoint of their end users. However, they also have several important disadvantages that we believe to limit their widespread deployment in practice.

First, most projections inherently introduce errors in terms of preserving distances and/or neighborhoods when projecting data from high-dimensional spaces to low-dimensional (2D or 3D) spaces. Unless such errors are clearly quantified and explained to the end user, interpreting projections may be misleading. We propose, in Chapters 3 and 4, a set of visual techniques that enhance multidimensional projections to make it easier for users to analyse their quality in terms of distance- and neighborhood-preservation. They improve on previous work related to this task by offering, among other contributions: (i) multiple correlated views (which can be easily switched or used in parallel) of the same projection/dataset, each showing a different perspective on the projection errors; (ii) new metrics for computing the errors, both for distance- and neighborhood-preservation, that address the problem from new points of view; (iii) a new multiscale visual presentation for projection errors that allows easier visual detection and reasoning about large compact groups of points in a projection than when using scatterplots; (iv) an approach to showing projection errors related to one-to-many and many-to-many point relationships, using edge bundling; and (v) the generalization of error-analysis techniques to groups of points instead of single points.

Secondly, projections create visualizations that show observations (and their similarities) well, but do not show the underlying data attributes. Unless this is done, users may dismiss multidimensional projections as being too abstract and/or far away from their original high-dimensional attribute space. The problem is even worse with 3D projections; while 2D projections may come with a reasonably rich set of explanatory techniques, the more accurate 3D projections do not offer such techniques. Unless this is done, users may choose to remain confined to using 2D projections, which may introduce significantly larger distance and neighborhood-preservation errors. To deal with these problems, we propose new attribute-based techniques for explaining projections with low-dimensional representations both in 3D and 2D. For 3D projections (Chapter 5), the new proposed techniques improve on previous work by offering, among other contributions: (i) ways of selecting the best viewpoints in the 3D space that can show combinations of variables of interest; (ii) generalizing biplot axes for nonlinear projection methods; and (iii) widgets for viewpoint navigation that help users to move smoothly between combinations of variables of interest. For the 2D projections (Chapter 6), the contributions concern (i) explaining local neighborhoods in a projection instead of global projection axes; (ii) automatic and implicit partition of the projection space into regions explained by the same attributes, which handles also projections without clear separations between points.

Finally, for multivariate datasets containing both scalar and relational attributes, or multivariate graphs, projections have been only marginally used as a visualization technique. Exploring projections as a tool for generating visualizations of such datasets is a potentially effective avenue for creating insightful displays of multivariate networks. The techniques proposed in Chapter 7 present (i) a framework for using distance-based multidimensional projections for the layout of graphs, which opens up the possibility of using any of the existing MDS techniques on the literature to create graph layouts; (ii) the creation of layouts based on a weighted mix of the network's attributes and distances, depending on the user's exploratory requirements; and (iii) the application of the projection-error metrics presented in the earlier chapters into the context of multivariate networks.

The remainder of this thesis is dedicated to exploring in details the above-mentioned research directions and improvements, which are centered on making multidimensional projections effective tools for the exploration of general-purpose multivariate datasets.

## Chapter 3

# Visualizing Distance Preservation

As outlined in Chapter 2, multidimensional projections are effective tools for exploring large sets of high-dimensional observations, especially when the questions of interest regard the detection of groups of similar observations or isolated outlier observations. However, as the same chapter has explained, projection techniques are challenged by the inability of faithfully preserving the so-called structure of the high-dimensional data, such as inter-point distances or point neighbors. If such aspects of the data structure are not well preserved – or, alternatively, if the lack of preservation is not clearly shown to the user, the visualization may convey wrong insights in the data.

In this chapter, we address the first part of the above challenge, namely measuring and displaying *distance preservation errors* in multidimensional projections. In detail, we present a set of metrics that aim to address the following questions for 2D projections:

- How is the projection error spread over the 2D space?
- How to find points that are close in 2D but far in $n$D?
- How to find points that are close in $n$D but far in 2D?
- How do the choice of projection algorithm and its parameter settings affect the above quality aspects?

To visualize the proposed metrics, we develop several space-filling techniques that visually scale to large datasets, offer a multiscale (or level-of-detail) view on the projection behavior, and do not require users to understand the internal formulation of dimensionality-reduction algorithm. We next use these visualizations to explore how five state-of-the-art dimensionality-reduction techniques behave, in terms of distance-preservation errors, when varying their parameters. Our endeavor of exploring projection errors is completed, next, by the work presented in Chapter 4, which addresses the related, but different, task of measuring and analysing neighborhood preservation errors for multidimensional projections.

## 3.1 Analysis Goals

Let us first recall the basic notations used to describe multidimensional projections. A multidimensional projection technique was modeled as a function $f : \mathbb{R}^n \times P \rightarrow \mathbb{R}^m$ that takes a dataset $D^n \subset \mathbb{R}^n$ and maps it to a lower dimensional dataset $D^m \subset \mathbb{R}^m$, $m < n$ (see Eqn. 2.1 and related text in Sec. 2.3). Here, $P$ indicates the space of parameters of the projection technique $f$, which depends on the specific details of the algorithm underlying $f$.

A projection $f$ should preserve the *structure* of the original space $\mathbb{R}^n$. This implies, as discussed in Chapter 2, a mix of distance and neighborhood preservations at various scales and happens at different rates for different datasets, projection algorithms, and parameter values. One important task for which projections are used is to detect and reason about groups of close points in $D^m$. Unless users are sure that such groups of points are indeed also close in the original high-dimensional dataset $D^n$, the interpretation of the projection result can be misleading [8]. Hence, for such tasks, the perceived precision (or, in other words, quality) of a projection is intrinsically linked to its ability to preserve distances between points [159].

Our exploration goal can be thus refined as follows: Given any dimensionality-reduction (DR), or projection, algorithm (Eqn. 2.1), we aim to show how distance preservation is affected by choices of parameter values in $P$, highlighting aspects that can adversely affect the interpretation of the projected point set $D^m$. To simplify the discourse, we next consider $m = 2$, and that projections are drawn as scatterplots – the most common option for DR visualization. We identify the following aspects of interest in the exploration of distance preservation in multidimensional projections:

**A. False neighbors:** Take a point $\mathbf{p}_i \in D^n$ and its 2D projection $\mathbf{q}_i = f(\mathbf{p}_i)$. A necessary condition for distance preservation is that *all* points $\mathbf{q}_j$ that are close to $\mathbf{q}_i$ (in 2D) should be projections of points $\mathbf{p}_j$ that are close to $\mathbf{p}_i$ (in $D^n$). If not, *i.e.* we have a $\mathbf{q}_j$ close to $\mathbf{q}_i$ for which $\mathbf{p}_j$ is not close to $\mathbf{p}_i$, the user wrongly infers from the projection that $\mathbf{p}_j$ is close to $\mathbf{p}_i$. We call such a point $j$ a *false neighbor*[1] of $i$.

**B. Missing neighbors:** The second necessary condition for distance preservation is that *all* $\mathbf{p}_j$ that are close to $\mathbf{p}_i$ (in $D^n$) project to points $\mathbf{q}_j$ that are close to $\mathbf{q}_i$ (in 2D). If not, *i.e.* we have a $\mathbf{p}_j$ close to $\mathbf{p}_i$ for which $\mathbf{q}_j$ is not close to $\mathbf{q}_i$, the user will underestimate the set of points similar to point $i$. We call such a point $j$ a *missing neighbor* of $i$.

**C. Groups:** A main goal of DR is to help users find groups of similar points, *e.g.* topics in a document set [95, 139] or classes of images in a database [55]. False and missing point neighbors generalize, for groups, to *false members* and *missing members* respectively. Given a group $\Gamma$ of closely projected points, we aim to find if *all* points in $\Gamma$ truly belong there (no false members), and if *all* points that belong to the topic described by $\Gamma$ do indeed project in $\Gamma$ (no missing members).

**D. Detail:** Aggregated local metrics such as [159, 8, 111, 79] can show, up to various extents, where missing or false neighbors occur. However, they do not directly show which are all such neighbors, for each projected point. Also, they do not explicitly address locating false and missing group members. We aim to provide interactive visual mechanisms to support these tasks on several levels of detail.

[1] It is important to note that, although our discourse here uses terms such as false neighbors and missing neighbors, the projection-error visualization techniques discussed in this chapter focus on showing *distance*-preservation errors and not *neighborhood*-preservation errors, the latter being the subject of Chapter 4. We employ here terms that use the 'neighbor' concept simply because this term is more compact, and arguably more illustrative, than alternative distance-related terms.

We next propose several visualization methods to address the analysis goals outlined in this section.

## 3.2 Visualization Methods

As a running example, we use the Locally Affine Multidimensional Projection (LAMP) technique as projection method, with the default parameter settings given in [95], and as input the well-known 19-dimensional Segmentation dataset with 2300 points from [116, 95, 138, 141]. In this dataset, each point describes a randomly drawn 3x3 pixel-block from a set of 7 manually segmented outdoor images, by means of 19 statistical image attributes, such as color mean, standard deviation, and horizontal and vertical contrast. The aim of analysing this multidimensional dataset is to see how well extracted features group over observations, and thereby derive insights that can further on help in the construction of automatic classifiers for natural images. For completeness, we note that the task of classifier construction is outside our scope – our goal is strictly the exploration of the capability of projection techniques in displaying the similarity of high-dimensional observations, and errors involved in this display.

### 3.2.1 Preliminaries

To quantify the distance preservation issues in Sec. 3.1, we first define the projection error of point $i$ *vs* a point $j \neq i$ as

$$e_{ij} = \frac{d^m(\mathbf{q}_i, \mathbf{q}_j)}{\max_{i,j} d^m(\mathbf{q}_i, \mathbf{q}_j)} - \frac{d^n(\mathbf{p}_i, \mathbf{p}_j)}{\max_{i,j} d^n(\mathbf{p}_i, \mathbf{p}_j)}. \tag{3.1}$$

We see that $e_{ij} \in [-1, 1]$. Negative errors indicate points whose projections are too close (thus, false neighbors). Positive errors indicate points whose projections are too far apart (thus, missing neighbors). Zero values indicate 'good' projections, which approximate optimally the distances in $D^n$.

Comparing Eqn. 3.1 with the aggregated normalized stress definition $\sigma$ (Eqn. 2.2), we see similarities but also several differences. First, both errors are essentially differences between the distances in $D^m$ and $D^n$ between data points – the more these distances are different, the higher is the error. However, our error $e_{ij}$ is not a strictly positive value as the (terms of the) stress function. This allows us to distinguish, on a more refined level, points that are projected too close from points that are projected too far away. Secondly, our error is normalized between $-1$ and $1$, which allows us to compare more easily different projections and/or different points in the same projection. In turn, both above aspects allow us to easily create several detail visualizations of projection errors. These are described next.

### 3.2.2 The Aggregated Error view

We first provide an overview of how the projection error spreads over an entire dataset, by computing for each point $i$ the aggregate error

$$e_i^{aggr} = \sum_{j \neq i} |e_{ij}|. \tag{3.2}$$

The value of $e_i^{aggr}$ gives the projection error of point $i$ with respect to *all* other points. Low values of $e^{aggr}$ show points whose projections can be reliably compared with most other projections in terms of assessing similarity. These are good candidates for representatives in multilevel projection methods [55, 144, 29, 139]. Large values of $e^{aggr}$ show points that are badly placed with respect to most other points. These are good candidates for manual projection optimization [142, 138].

Fig. 3.1 (a) shows $e^{aggr}$ by color mapping its value on the 2D projected points, using a blue-yellow-red diverging colormap [74]. Brushing and zooming this image allows inspecting $e^{aggr}$ for individual points. However, given our goal of providing an overview first, we are actually not interested in *all* individual $e^{aggr}$ values, but rather to (a) find compact areas in the projection having similar $e^{aggr}$ values, (b) find outlier $e^{aggr}$ values in these areas (if any), and (c) see how $e^{aggr}$ globally varies across the projection. For this, we propose an image-based, space-filling visualization, as follows. Denote by $DT(\mathbf{x} \in \mathbb{R}^2) = \min_{\mathbf{q} \in D^m} \|\mathbf{q} - \mathbf{x}\|$ the so-called distance transform of the 2D point cloud $D^m$ delivering, for any screen pixel $\mathbf{x}$, its distance to the closest point in $D^m$ [39]. We then compute $e^{aggr}$ at every screen pixel $\mathbf{x}$ as

$$e^{aggr}(\mathbf{x}) = \frac{\sum_{\mathbf{q} \in N_\epsilon(\mathbf{x})} \exp\left(-\frac{\|\mathbf{x}-\mathbf{q}\|^2}{\epsilon^2}\right) e^{aggr}}{\sum_{\mathbf{q} \in N_\epsilon(\mathbf{x})} \exp\left(-\frac{\|\mathbf{x}-\mathbf{q}\|^2}{\epsilon^2}\right)} \tag{3.3}$$

with

$$\epsilon = DT(\mathbf{x}) + \alpha. \tag{3.4}$$

Here, $N_\epsilon(\mathbf{x})$ contains all projections in $D^m$ located within a radius $\epsilon$ from $\mathbf{x}$. We next draw $e^{aggr}(\mathbf{x})$ as a RGBA texture, where the color components encode $e^{aggr}(\mathbf{x})$ mapped via a suitable color map, and the transparency $A$ is set to

$$A^{aggr}(\mathbf{x}) = \begin{cases} 1 - \frac{DT(\mathbf{x})}{\alpha}, & \text{if } DT(\mathbf{x}) < \beta \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

For $\alpha = 1, \beta = 1$, we obtain the classical colored scatterplot (Fig. 3.1 (a)). For $\alpha = 1, \beta > 1$, the space between projections is filled, up to a distance $\beta$, by the $e^{aggr}$ value of the closest data point. For $\alpha = 1, \beta = \infty$, we obtain a Voronoi diagram of the projections with cells colored by their $e^{aggr}$ values. This does not change the $e^{aggr}$ data values, but just displays them on larger spatial extents than individual pixels, making them easier to see, similarly to the use of Voronoi cells to show attributes in multidimensional projections in [24, 111]. This creates visualizations identical to those obtained by drawing scatterplots with point radii equal to $\beta$, without having the issues

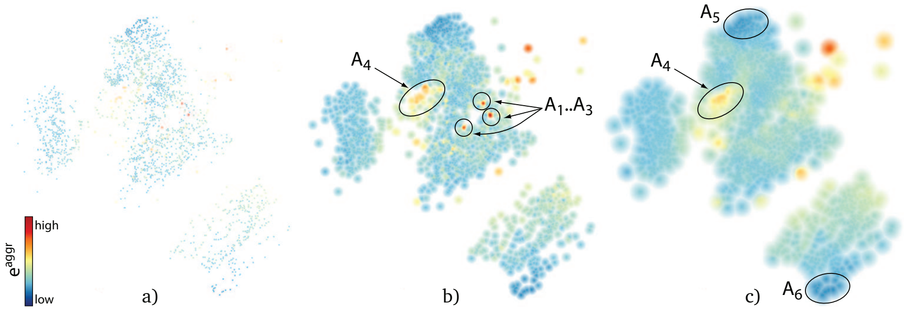


Figure 3.1: Aggregate error view, several levels of detail: (a) $\alpha = 1, \beta = 1$. (b) $\alpha = 5, \beta = 5$. (c) $\alpha = 20, \beta = 20$ pixels (see Sec. 3.2.2).

created by overlapping points. For $\alpha > 1, \beta > 1$, the result is similar to a smooth Shepard interpolation where the kernel size $\epsilon$ is given by the *local* point density. The parameter $\alpha \geqslant 0$ controls the *global* level-of-detail at which we visualize $e^{aggr}$: Small values show more detail in dense point zones, but also emphasize small-scale signal variations that are less interesting. Larger $\alpha$ values create a smoother signal where coarse-scale error patterns are more easily visible.

Figs. 3.1 (b,c) show the aggregate error for the Segmentation dataset for various values of the parameters $\alpha$ and $\beta$. Here, $e_{ij} \in [-0.67, 0.35]$. The error range already tells that we have poorly projected points, but does not tell where these are. In Fig. 3.1 (b), with low values for both $\alpha$ and $\beta$, we see that $e^{aggr}$ is relatively smoothly distributed over the entire projection. However, we see three small red spots $A_1..A_3$. These are high-error outlier areas, which indicate points that are badly placed with respect to *most* other points. We also see a relatively high error area $A_4$ of larger spatial extent. Increasing both $\alpha$ and $\beta$ produces a simplified visualization (Fig. 3.1 (c)). Larger $\beta$ values fill in the gaps between points. Larger $\alpha$ values eliminate outlier regions whose spatial extent is smaller than $\alpha$, such as the three small outlier areas $A_1..A_3$, but $A_4$ remains visible, since it is larger than $\alpha$. We now also notice, better than in Fig. 3.1 (b), that the bottom and top areas ($A_5, A_6$) in the projection have dark blue values, with a significantly lower error than the rest of the projection.

Our image-based results are slightly reminiscent of the dense projection-precision-score (*pps*) maps of Schreck *et al.* [159]. Differences exist, however. First, our $e_i^{aggr}$ is a *global* metric, that tells how point $i$ is placed with respect to all other points, whereas the *pps* metric characterizes *local* neighborhoods. Interpolation-wise, our technique (used with $\alpha = 1, \beta = \infty$) delivers the same Voronoi diagram as Schreck *et al.*, which is also identical to the space partitioning of the point-based Voronoi diagrams in [8, 111]. The data being mapped is, however, different: Our $e^{aggr}$ shows the sum of distance compression and stretching, whereas the techniques in [8, 111] treat these two quantities separately. In the next sections, we show how we split our aggregated insight into separate insights. Further on, both Schreck *et al.* and our method use smoothing to remove small-scale noise from such maps. However, whereas Schreck *et al.* uses a constant-radius smoothing kernel, which blurs the image equally strongly everywhere, we use, as explained, a variable-radius kernel controlled by local density, which preserves better detail in non-uniform point clouds (scatterplots).

### 3.2.3 The False Neighbors view

While it is useful to assess the error distribution and find badly *vs* well-projected point groups, the aggregate error view does not tell us if the error is due to false neighbors, missing neighbors, or both. Let us first consider the false neighbors (case **A**, Sec. 3.1). To visualize these, we create a Delaunay triangulation of the projected point cloud that gives us the closest neighbors of each projected point in all directions, *i.e.*, the most important false-neighbor candidates for that point. To each edge $E_k, 1 \leqslant k \leqslant 3$ of each triangle $T$ of this triangulation, with vertices being the points $\mathbf{q}_i$ and $\mathbf{q}_j$ of $D^m$, we assign a weight $e_k^{false} = |\min(e_{ij}, 0)|$, *i.e.*, consider only errors created by false neighbors. Next, we interpolate $e^{false}$ over all pixels $\mathbf{x}$ of $T$ by using

$$e^{false}(\mathbf{x}) = \frac{\sum_{1 \leqslant k \leqslant 3} \frac{1}{d(\mathbf{x}, E_k)\ \|E_k\|} e_k^{false}}{\sum_{1 \leqslant k \leqslant 3} \frac{1}{d(\mathbf{x}, E_k)\ \|E_k\|}} \tag{3.6}$$

where $d(\mathbf{x}, E)$ is the distance from $\mathbf{x}$ to the edge $E$ and $\|E\|$ is the length of the edge. Similarly to the aggregated error, we construct and render an image-based view for $e^{false}$ as a RGBA texture. In contrast to the aggregated error, we use here a heated body colormap [74], with light hues showing low $e^{false}$ values and dark hues showing high $e^{false}$ values. This attracts the attention to the latter values, while pushing the former ones into the background. The transparency $A$ is given by

$$\begin{aligned} A^{false}(\mathbf{x}) = A^{aggr}(\mathbf{x}) \Bigg(1 - \frac{1}{2} \bigg(\min\left(\frac{DT_T(\mathbf{x})}{DT_C(\mathbf{x})}, 1\right) + \\ \max\left(1 - \frac{DT_C(\mathbf{x})}{DT_T(\mathbf{x})}, 0\right)\bigg)\Bigg) \end{aligned} \tag{3.7}$$

where $DT_T(\mathbf{x}) = \min(d(\mathbf{x}, E_1), d(\mathbf{x}, E_2), d(\mathbf{x}, E_3))$ is the distance transform of $T$ at $\mathbf{x}$, $DT_C(\mathbf{x})$ is the distance from $\mathbf{x}$ to the barycenter of $T$, and $A^{aggr}$ is given by Eqn. 3.5. The same technique is used in a different context to smoothly interpolate between two 2D nested shapes [153], to which we refer for further (simple) implementation details. The combined effect of Eqns. 3.6 and 3.7 is to slightly thicken, or smooth out, the rendering of the Delaunay triangulation. Note that this interpolation does *not* change the actual values $e_k^{false}$ rendered on the triangulation edges. The distance-dependent transparency ensures that data is shown only close to the projection points.

Fig. 3.2 shows the false neighbors for the Segmentation dataset. Several aspects are apparent here. First, the rendering is similar to a blurred rendering of the Delaunay triangulation of the 2D projections colored by $e^{false}$, showing how each point relates to its immediate neighbors. Light-colored edges show true neighbors, while dark edges show false neighbors. Since edges are individually visible, due to the transparency modulation (Eqn. 3.7), we can see both the true and false neighbors of a point separately. The smooth transition between opaque points (on the Delaunay edges) and fully transparent points (at the triangles' barycenters) ensures that the resulting image is continuous and

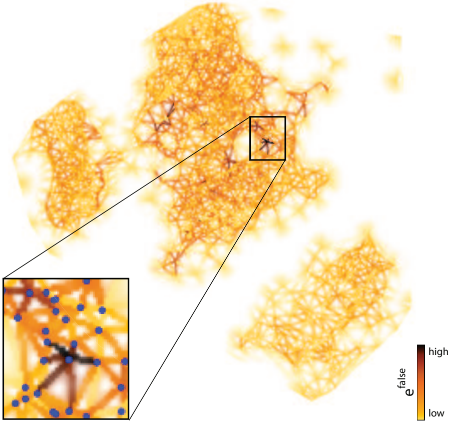


Figure 3.2: False neighbors view (see Sec. 3.2.3).

easier to follow at various screen resolutions than a Delaunay triangulation rendered with pixel-thin edges, as our edges appear slightly thicker.

In Fig. 3.2, two error-related aspects are visible. First, we see an overall trend from light to dark colors as we go further from the projection's border towards the projection center. This confirms the known observation on DR methods that points projected near the border tend to be more accurate, since there is more freedom (and space) to place them. In contrast, projections falling deep inside the resulting point cloud tend to have more false neighbors, because the DR algorithm has there less space to shift points around to accommodate all existing distance constraints. Intuitively, we can think of this phenomenon as a 'pressure' which builds up within the projected point set from its border inwards. We shall see more examples of this phenomenon in Sec. 3.3. Secondly, we see a few small-scale dark outliers. Zooming in Fig. 3.2, we see that these are points connected by dark edges to most of their closest neighbors in a star-like pattern. Clearly, false neighbors exist here. These can be either the star 'center' or the tips of its branches. However, we also see that these tips have only one dark edge. Hence, they are too closely positioned to the star center only, and not to their other neighbors. Since the tip points are all positioned well with respect to their neighbors (except the star center), and the center point is positioned too closely with respect to all its direct neighbors, we can conclude that too little space was offered in the projection to the center point, or in other words that the center point is a false neighbor of its surrounding points.

The false neighbors view is related to Aupetit's segment compression view, where the shortening of inter-point distances due to projection is visualized [8]. The underlying metrics, *i.e.* our $e_{ij}$ (Eqn. 3.1) and $m_{ij}^{distor}$ ([8], Sec. 3.2) are similar, up to different normalizations. However, the proposed visualizations are quite different. Aupetit uses so-called 'segment Voronoi cells' (SVCs). SVCs essentially achieve piecewise-constant ($C^{-1}$) interpolation of the values $e_k^{false}$, defined on the edges $E_k$ of each Delaunay triangle $T$, over $T$'s area, by splitting $T$ in three sub-triangles using its barycenter. In contrast, our interpolation (Eqn. 3.6) is $C^{\infty}$ over $T$. Also, our triangles are increasingly transparent far away from their edges (Eqn. 3.7). Comparing our results (*e.g.* Figs. 3.2, 3.9 (a,d,g)) with SVCs (*e.g.* Figs. 7 (d), 12 (c) in [8]), we observe that SVCs exhibit several spurious elongated Voronoi cells that do not convey any information. Such cells do not exist in our visualization due to the transparency blending. Also, we argue that the artificial SVC edges linking projected points with Delaunay triangulation barycenters do not convey any information, but only make the visualization more complex. Such edges do not exist in our visualization due to our continuous interpolation.

### 3.2.4 The Missing Neighbors view

Besides false neighbors, projection errors (and subsequent misinterpretations) can also be caused by missing neighbors (case **B**, Sec. 3.1). Visualizing this by a space-filling method like for the aggregate error or false neighbors is, however, less easy. Given a projected point $\mathbf{q}$, its missing neighbors can be anywhere in the projection, and are actually by definition far away from $\mathbf{q}$. To locate such neighbors, we would need to visualize a many-to-many relation between far-away projected points.

We first address this goal by restraining the question's scope: Given a *single* point $\mathbf{q}_i$, show which of the other points $D^m \backslash \mathbf{q}_i$ are missing neighbors for $\mathbf{q}_i$. For this, we first let the user select $\mathbf{q}_i$ by means of direct brushing in the visualization. Next, we compute the error $e_i^{missing} = \max_{j \neq i}(e_{ij}, 0)$, *i.e.*, the degree to which $\mathbf{q}_j$ is a missing neighbor for $\mathbf{q}_i$, and visualize $e^{missing}$ by the same technique as for the aggregated error (Sec. 3.2.2).

Fig. 3.3 shows this for the Segmentation dataset, using the same heat colormap as in Fig. 3.2. In Figs 3.3 (a,b), we selected two points deep inside the central, respectively the lower-right point groups in the image. Since Figs. 3.3 (a,b) are nearly entirely light-colored, it means that these points have few missing neighbors. Hence, the 2D neighbors of the selected points are truly *all* the neighbors that these points have in $n$D. In Figs. 3.3 (c,d), we next select two points located close to the upper border of the large central group and the left border of the left group respectively. In contrast to Figs. 3.3 (a,b), we see now an increasingly darker color gradient as we go further from the selected points. This shows that points far away from these selections are actually projected *too* far, as they are actually more similar than the projection suggests. This is a known (but never visualized as such) issue of many DR methods, which have trouble in embedding high-dimensional manifolds in 2D: points close to the embedding's *border* are too far away from other points in the projection. Another interesting finding is that the color-coded Figs. 3.3 (c,d) do not show a *smooth* color gradient: We see, especially in Fig. 3.3 (c) that the colors appear grouped in several 'bands', separated by discontinuities. In other words, the projection method suddenly increases the error as we get over a certain maximal 2D distance.

The missing neighbors view is related to the proximity view of Aupetit [8]. In both views, a point $i$ is selected and a scalar value, related to this selection, is plotted at all other points $j \neq i$. For Aupetit, this is the distance $m_j^{prox} = d^n(\mathbf{p}_i - \mathbf{p}_j)$ (normalized by its maximum). For us, it is the error $e_j^{missing}$. Both the distance and $e^{missing}$ have, in general, the tendency to be small at points $j$ close in 2D to the selected point $i$, and increase farther off from point $i$. However, the two quantities are different and serve different purposes. Visualizing $m^{prox}$ is useful in finding points located within some

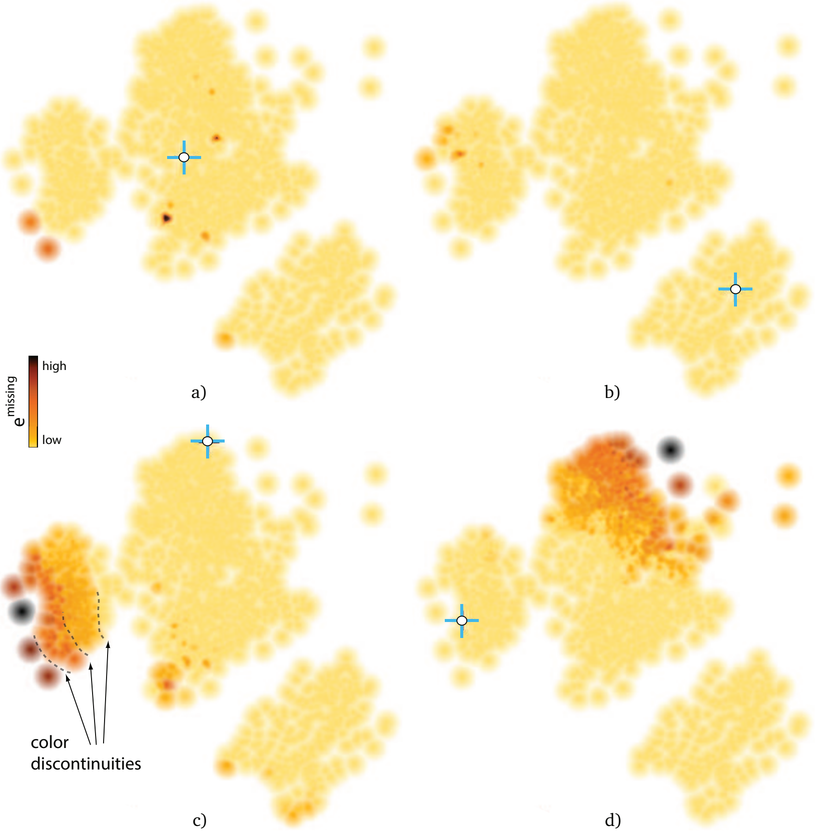


Figure 3.3: Missing neighbors view for different selected points. Selections are indicated by markers (see Sec. 3.2.4).

distance to the selection $i$. Finding projection errors is only *implicitly* supported, as these appear as non-monotonic variations in the $m^{prox}$ signal. In contrast, $e^{missing}$ specifically emphasizes points projected too far, rather than conveying the absolute distance. Thus, our visualization helps locating projection errors rather than assessing proximity.

### 3.2.5 The Missing Neighbors Finder

Although providing details for single points, the views in Sec. 3.2.4 cannot show missing neighbors for an entire dataset. We address this goal by a different method, as follows. Consider all positive values of $e_{ij}$. By definition, these give all point-pairs which are projected too far away. We sort these values decreasingly, and select the largest $\phi$ percent of them, where $\phi$ is a user-provided value. The selected values give the point pairs which are worst placed in terms of overestimating their true similarity. We next construct a graph $G = (V, E)$ whose nodes $V$ are the projected points $\mathbf{q}_i$ present in such point pairs, and edges $E$ indicate the pairs, with $e_{ij}$ added as edge weights. Next, we draw

$G$ using the KDEEB edge bundling technique [85], which provides robust, easy to use, and real-time bundling of graphs with tens of thousands of edges on a modern GPU. We color the bundled edges based on their weight using a grayscale colormap (with white mapping low and black mapping high weights), and draw them sorted back-to-front on weight and with an opacity proportional to the same weight. The most important edges thus appear atop and opaque, and the least important ones are at the bottom and transparent.

Fig. 3.4 shows this visualization, which we call the *missing neighbors finder*, with bundles that connect a single selected point with its most important missing neighbors (bundles connecting multiple points are discussed later on). The background images show $e^{missing}$ (Sec. 3.2.4). Dark bundle edges attract attention to the most important missing neighbors. For the selected points in images (a) and (b), we see that there are only very few and unimportant missing neighbors (few half-transparent edges). For the selected points in images (c) and (d), the situation is different, as the bundles are thicker and darker. Bundle fanning shows the *spread* of missing neighbors for the selected points: In image (c), these are found mainly in the left point group, with a few also present in the lower part of the central group. In contrast, all missing neighbors of the point selected in image (d) are at the top of the central group.

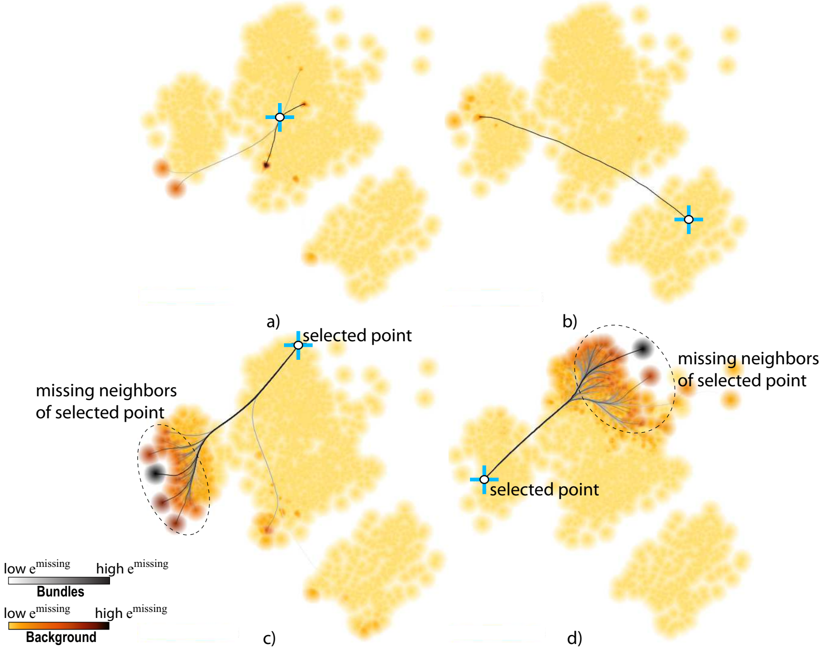
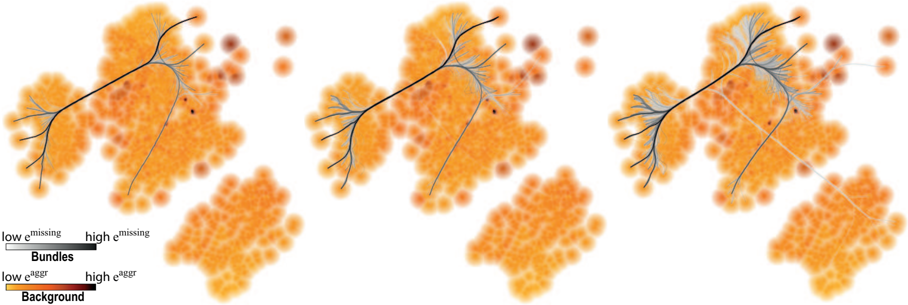


Figure 3.4: Missing neighbors finder view for four selected points. Selections are indicated by markers (see Sec. 3.2.5).

The main added value of the missing neighbors finder appears when we visualize the many-to-many relations given by all projected points. Fig. 3.5 shows this result for three values of $\phi$ for the Segmentation dataset. The background shows now the aggregated error ($e^{aggr}$, Sec. 3.2.2). We color bundles from black for largest error $e_{ij}$ to white for largest error above the user-provided parameter $\phi$. Image (a) shows the $\phi = 1\%$ worst missing-neighbor point-pairs. These link the top-right area of the central group with the left frontier of the left group. Adding more missing neighbor pairs to the view (image (b), $\phi = 3\%$) strengthens this impression. Adding even more missing neighbor pairs (image (c), $\phi = 20\%$) reveals additional missing-neighbor pairs between the two areas indicated above (light gray parts of thick top bundle), and also brings in a few missing neighbors between these areas and the lower-right point group (light gray thin bundle going to this group). Nearly all bundles appear to connect point pairs located on the *borders* of the projection. This strengthens our hypothesis that such point pairs are challenging for the LAMP projection, which we noticed using the interactive missing neighbors view (Sec. 3.2.4). However, as compared to that view, the bundled view shows all such point pairs in a single go, without requiring user interaction.



a) ϕ=1%

b) ϕ=3%

c) ϕ=20%

Figure 3.5: Missing neighbors finder view, all point pairs, for different $\phi$ values (see Sec. 3.2.5).

### 3.2.6 The Group Analysis views

As outlined in Sec. 3.1, the false and missing neighbors issues for individual points become, at group level, the problems of false and missing group members respectively. We next propose two visualizations that assist in finding such issues.

First, let us refine the notion of a group. Given the tasks in Sec. 3.1 (C), a group $\Gamma \subset D^m$ is a set of projected points which form a *visually* well-separated entity. When users see points in a group, they understand that these share some commonality, but are different from points in other groups. In the LAMP projection of our Segmentation dataset, we see three such groups (Figs. 3.1-3.5). Group perception is, obviously, subject to many factors such as user preferences and level-of-detail at which one focuses. However, once a user has established which are the groups (s)he sees in a visualization, the false and missing membership issues become relevant.

We allow users to select groups in a given projection by several mechanisms: direct interactive selection, mean-shift clustering [36], and upper thresholding of the point density [52]. Other user-controlled methods can be used if desired, *e.g.*, $K$-means or hierarchical agglomerative clustering, *e.g.* [92, 91]. The actual group selection mechanism is further of no importance to our visualization method. We next render each obtained group $\Gamma = \{\mathbf{q}_i\}$ by the shaded cushion technique in [182] as follows. First, we compute a density map $\rho(\mathbf{x}) = \sum_{\mathbf{q}\in\Gamma} K(\mathbf{x} - \mathbf{y})$, where $K$ is an Epanechnikov kernel of width equal to the average inter-point distance $\delta$ in $\Gamma$, following [36]. Next, we compute a threshold-set $\Gamma_\delta$ of $\rho$ at level $\delta$, and its distance transform $DT_{\Gamma_\delta}$. Finally, we render a RGBA texture over $\Gamma_\delta$, where we set the color a fixed hue (light blue in our case) and the transparency A to $\sqrt{DT_{\Gamma_\delta}}$ (following the approach in [182] which shows that such a transparency profile creates naturally-looking cushions).

Having now groups both as a data structure and also shown in the visualization, we adapt the missing neighbors and finder techniques (Secs. 3.2.4, 3.2.5) to show missing group members. For this, we compute a value

$$e_\Gamma^{missing}(\mathbf{q}_i) = \begin{cases} \min_{\mathbf{q}_j\in\Gamma}(e_{ij}) & \text{if } \mathbf{q}_i \notin \Gamma \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

at each projected point $\mathbf{q}_i$, and visualize $e_\Gamma^{missing}$ using the same technique as for missing neighbors.

Fig. 3.6 (a,b) show two missing group members views. The shaded cushions show the three groups identified in our Segmentation dataset. Several points fall outside of all groups. This is normal, in general, *e.g.* when the user cannot decide to which group to associate a point. In image (a), we select the bottom group $\Gamma_{bottom}$. The underlying color map shows now $e_{\Gamma_{bottom}}^{missing}$, (Eqn. 3.8). All points appear light yellow. This means that, with respect to $\Gamma_{bottom}$ *seen as a whole*, no points are projected too far, so $\Gamma_{bottom}$ has no missing members. In image (b), we do the same for the left group $\Gamma_{left}$. The image now appears overall light yellow, except for a small dark-red spot in the upper-right corner of the central group $\Gamma_{center}$. Here are a few points which are placed too far from any point in $\Gamma_{left}$. These are highly likely to be missing members of $\Gamma_{left}$. To obtain more insight, we now use the bundle view in Sec. 3.2.5, with two changes. First, we build only bundles that have an endpoint in the selected group. Secondly, we consider all edges rather than showing only the most important ones. Image (c) shows the bundle view for $\Gamma_{bottom}$. We see only a few bundled edges, ending at a small subset of the points in $\Gamma_{bottom}$. This strengthens our hypothesis that there are no points outside $\Gamma_{bottom}$ which should be placed closer to *all* points in $\Gamma_{bottom}$ – or, in other words, that $\Gamma_{bottom}$ has no missing members. Image (d) shows the bundled view for $\Gamma_{left}$. The bundle structure tells us that the top-right part of $\Gamma_{center}$ contains many missing neighbors of $\Gamma_{left}$. In particular, we see dark bundle edges that connect to dark-red points. This is a strong indication that these points can indeed be missing members of $\Gamma_{left}$. For a final assessment, the user can interactively query the discovered points' details (attribute values) and, depending on these, finally decide if these points are missing group members or not.

### 3.2.7 The Projection Comparison view

Consider running the same DR algorithm with two different parameter sets, or projecting a dataset by two different DR algorithms. How to compare the results from the viewpoint

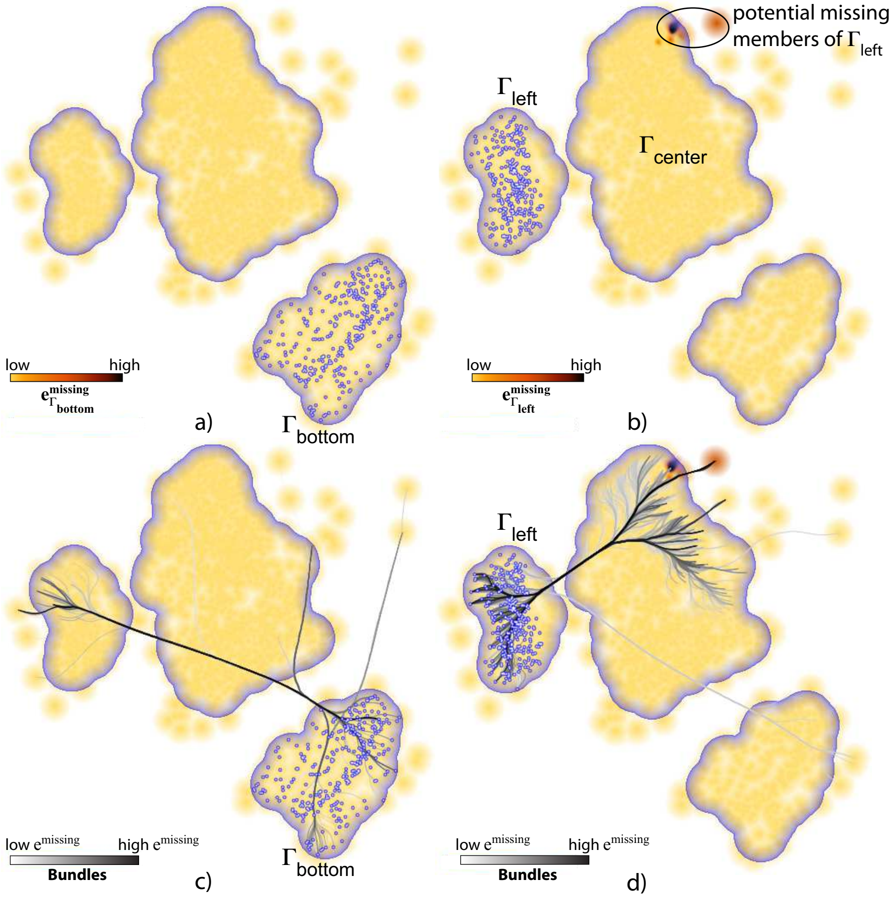


Figure 3.6: Missing members for two point groups. Points in the selected groups are drawn as marked (see Sec. 3.2.6).

of distance preservation? Subsequent questions are: Which points that were (correctly) placed close to each other in one projection are now 'pulled apart' in the other projection? Do the two projections deliver the same groups of points? Understanding all these aspects is crucial to further using the respective projections for interpreting the original high-dimensional data. Indeed, if we were able to see that specific point-groups are placed at significantly different locations in the 2D space by different projection techniques and/or parameter settings, this would mean that great care should be invested in assessing the respective tools (projection techniques and/or parameter settings) *prior to* interpreting the projection results. If, in contrast, the main point-groups in the projection would be the same with respect to technique and/or parameter settings, this would mean that one could use the respective projection tools very much like agnostic 'black boxes' further on.

To answer such questions, we propose the *projection comparison* view. The view reads two projections $D_1^m$ and $D_2^m$ of the same input dataset $D^n$. For each point-pair

($\mathbf{q}_i^1 \in D_1^m, \mathbf{q}_i^2 \in D_2^m$), we compute a displacement

$$e_i^{disp} = \frac{\|\mathbf{q}_i^1 - \mathbf{q}_i^2\|}{\max_i \|\mathbf{q}_i^1 - \mathbf{q}_i^2\|}. \tag{3.9}$$

We next build a graph whose nodes are points in $D_1^m \cup D_2^m$. Edges relate point pairs ($\mathbf{q}_i^1 \in D_1^m, \mathbf{q}_i^2 \in D_2^m$), and have the values $e^{disp}$ as weights. We visualize this graph via edge bundling, as for the missing neighbors finder (Sec. 3.2.5).

Fig. 3.7 (a) shows a view where we compare the Segmentation dataset projected via LAMP (red points, $D_1^m$) and LSP (green points, $D_2^m$). The two projections are quite similar, since red and green points occur together in most cases. However, this image does not tell if the two projections create the same *groups* of points, since we do not know how red points match the green ones. Fig. 3.7 (b) shows the projection comparison view for this case. We immediately see a thin dark bundle in the center: This links corresponding points which differ the most in the two projections. Correlating this with image (a), we see that LSP decided to place the respective points at the bottom ($A_{LSP}$) of the central group, while LAMP moved *and* also spread out these points to the top ($A_{LAMP}$). However, points around the locations $A_{LSP}$ and $A_{LAMP}$ do not move much between the two projections, as we see only light-colored bundles around these locations, apart from the dark bundle already discussed. Hence, the motion of these points indicates a neighborhood problem in one or both of the projections. Indeed, if *e.g.* the points in $A$ were correctly placed by LAMP (into $A_{LAMP}$), then the decision of LSP to move the point-group $A$ all the way up in the visualization (to $A_{LSP}$) should also have moved the *neighbors* of $A_{LAMP}$. Since this does not happen, $A_{LSP}$ cannot be close to the same points that $A_{LAMP}$ was. A similar reasoning applies if we consider that $A_{LSP}$ is correct – it then follows that $A_{LAMP}$ cannot be correctly placed with respect to its neighbors.

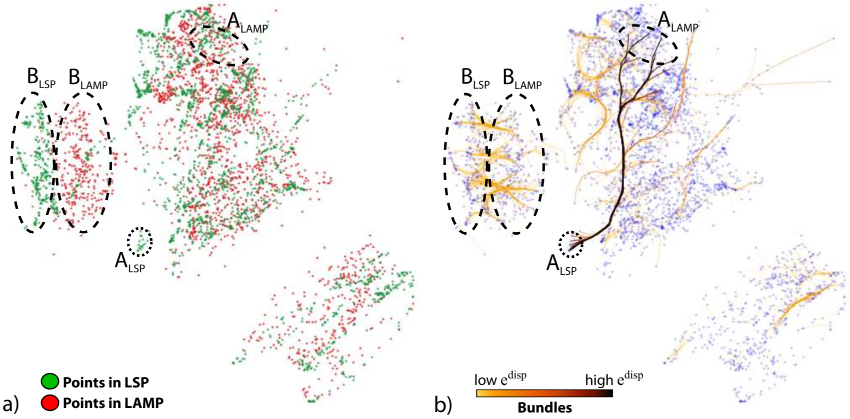


Figure 3.7: Comparison of two projections. (a) LAMP (blue) and LSP (red) points. (b) Bundles show corresponding point groups in the two projections (see Sec. 3.2.7).

Apart from this salient dark-colored bundle, we see many shorter and light-colored bundles. These show smaller-scale displacements between the two projections. For instance, we see how the red points at the right of the left group ($B_{LAMP}$) are moved to the left ($B_{LSP}$) of the same group. As these bundles fan out relatively little, do not have many crossings, and they are short, it means that $B_{LSP}$ is almost a *translation* to the left of $B_{LAMP}$, so the two projections depict the same structure of the left group. Also, we do not see any bundle exiting this left group. This means that both LAMP and LSP keep all points in this group together. Finally, in the bottom-right group we see just a very few short light-colored bundles. Most points in this group do not have any bundles connected to them. This means that $e^{disp}$ for these points is very small (yielding thus very short, nearly transparent, bundles). From this, we infer that LAMP and LSP produce very similar layouts for this group. If users are interested only to spot the most salient differences between two projections, and want to ignore such small-scale changes, this can be easily obtained by mapping $e_i^{disp}$ to bundle-edge transparency.

### 3.2.8 Usage scenario

Considering that the user is offered quite a few different views to analyse projection errors, each with specific features and goals, the next question arises: How to put all these views together to form a coherent *usage scenario* for a common analysis task? Below we propose such a usage scenario. The view names herein refer to the respective techniques presented earlier in this section.

**Step 1:** Start with the *Aggregated Error* view. This shows an overview of the error at all points, without a distinction between false or missing neighbors. Next, check if (a) there are regions or groups with substantial errors or (b) the overall error is low. Case (b) indicates that the projection is quite good and that nothing else needs to be improved. In case (a), continue with steps 2, 3, and 4.

**Step 2:** The *Missing Neighbors Finder* view can be enabled and disabled freely over the *Aggregated Error* view to show the most important missing neighbors between all points. The user should notice now whether this view shows bundles having high error values (*i.e.* dark-colored). If so, there are important missing neighbors between the groups connected by such bundles. These groups must be further analysed with the *Group Analysis Views*. If not, *i.e.* the bundles are colored (light) gray, this tells that the projection is good and, although there are missing neighbors, they are in a low error range and should not threaten the projection interpretation.

**Step 3:** Points, groups or regions found problematic in steps 1 and 2 are now analysed in more detail using the *False Neighbors* and *Missing Neighbors* views. For groups detected in step 1 the most important thing is to find out exactly what kind of error is present: Are they (a) wrongly placed with respect to each other and other close points (false neighbors) or (b) in relation to far away points that should be closer (missing neighbors)? For groups detected in step 2, the error is already identified from the beginning: They have a high rate of missing neighbors. In this case, the question to be answered is: Which points are exactly the problematic ones inside the detected groups, or where exactly do the relations (bundle edges) with the highest errors start and end from? By using these two views, the user should be able to establish exactly which are the more problematic points (or groups), and what kind of error these have.

**Step 4:** Knowing now where exactly errors occur, we consider the next questions: (1) Are such errors really a problem? (2) Do they show unexpected results related to how the projection should work with the provided data? (3) Are the problematic points important for the analysis task at hand? If questions (1-3) all answer 'no', then we have a good projection for our data and analysis task, and our analysis stops. If any question (1-3) answers yes, then the user must improve the projection of problematic points, as follows. If the user is a projection designer testing the accuracy of a new method, (s)he should go back to the algorithm and use the new insight gotten from this analysis to improve that algorithm. If the user has no access to the projection implementation, the solution is to re-execute the analysis from step 1 with either (i) a new projection algorithm that might better fit the specific data and task; or (ii) a new set of parameters for the same algorithm. The new results can be compared with the old ones to determine if the errors have decreased or if the errors moved into a new region where they are not as important for the task at hand. For the second task, the *Projection Comparison View* can be used.

## 3.3 Applications

We now use our views to study several projections for several parameter settings – thus, to explore the space $P$ that controls the creation of a DR projection. First, we present the datasets used (Sec. 3.3.1), the studied projection algorithms (Sec. 3.3.2), and their parameters (Sec. 3.3.3). Next, we use our views to explore the considered parameter settings (Secs. 3.3.4, 3.3.5).

### 3.3.1 Description of datasets

Apart from the Segmentation dataset used so far, we consider the following datasets:

**Freephoto:** contains 3462 images grouped into 9 unbalanced classes [58]. For each image, we extract 130 BIC (border-interior pixel classification) features. Such features are widely used in image classification tasks [173].

**Corel:** composed of 1000 photographs that cover 10 specific subjects. Similarly to the Freephoto dataset, we extract for each image a vector of 150 SIFT descriptors [115].

**News:** contains 1771 RSS news feeds from BBC, CNN, Reuters and Associated Press, collected between June and July 2011. The 3731 dimensions were created by removing stopwords, employing stemming and using term-frequency-inverse-document-frequency counts. We manually classified the data points based on the perceived main topic of the news feed resulting in 23 labels. Given the imprecision of the manual classification and the restriction to have one topic per point, the labels are unbalanced for a number of points. Also, for other points (with different labels), we can still have a high similarity of content.

**Sourceforge:** This publicly available dataset contains 24 software metrics computed on 6773 open-source C++ software projects from the sourceforge.net website [121]. Metrics include classical objet-oriented quality indicators such as coupling, cohesion, inheritance depth, size, complexity, and comment density [107], averaged for all source code files within a project.

### 3.3.2 Description of projections

We detail next the projection algorithms whose parameter spaces we will next study. We chose these particular algorithms based on their availability of documented parameters, scalability, genericity, presence in the literature, and last but not least availability of a good implementation.

**LSP:** The Least Squares Projection [139] uses a force-based scheme to first position a subset of the input points, called control points. The remaining points in the neighborhood of the control points are positioned using a local Laplace-like operator. Overall, LSP creates a large linear system that is strong in local feature definition. LSP is very precise in preserving neighborhoods from the $n$D space to the 2D space.

**PLMP:** The Part-Linear Multidimensional Projection (PLMP) [141] addresses computational scalability for large datasets by first constructing a linear mapping of the control points using the initially force-placed control points. Next, this linear mapping is used to place the remaining points, by a simple and fast matrix multiplication of the feature matrix with the linear mapping matrix.

**LAMP:** Aiming to allow more user control over the final layout, the Local Affine Multidimensional Projection (LAMP) [95] provides a user-controlled redefinition of the mapping matrix over a first mapping of control points. LAMP also works by defining control points, which are used to build a family of orthogonal affine mappings, one for each point to project. LAMP has restrictions regarding the number of dimensions against the number of points. Also, LAMP cannot directly work with distance relations, *i.e.*, it needs to access the $n$D point coordinates. However, LAMP is very fast, without compromising the precision reached, for instance, by LSP. Both LSP and LAMP can be controlled by a number of parameters, such as the control point set.

**Pekalska:** Another class of projection techniques works with optimization strategies. These are, in general, quite expensive computationally. To improve speed, Pekalska *et al.* [144] first embeds a subset of points in 2D by optimizing a stress function. Remaining points are placed using a global linear mapping, much like LAMP and LSP.

**ISOMAP:** The ISOMAP technique [184] is an extension of classical Multidimensional Scaling (MDS) that aims to capture nonlinear relationships in the dataset. ISOMAP replaces the input distance between point pairs by an approximation of the geodesic distance given by the shortest path on a graph created connecting neighbor points in the original space with the original distance as weight. The final 2D coordinates are computed via a conventional MDS embedding with calculations of eigenvalues over the distance relations of the previous step.

### 3.3.3 Description of parameters to analyse

Most techniques that initially project control points use a simplified iterative force-based algorithm, such as the one of Tejada *et al.* [179]. The number of *iterations* of force-based placement influences the control points' positions, and is, thus, a relevant parameter. LSP control points are typically the centroids of clusters obtained from a clustering of the input dataset. The *number of control points* is thus a second relevant parameter for LSP. To position points in the neighborhood of a given control point, LSP solves a linear system for that neighborhood. The neighborhood size (*number of neighbors*) is a third relevant parameter.

In LAMP, the affine mappings are built from a neighborhood of control points. The size of the control point set used to build the mapping, expressed as a *percentage* of the size of the control point set, is the main parameter here. The choice of control points and the choice of the initial projection of the control points are also parameterizable, just as for LSP, PLMP, and Pekalska. However, in LAMP, these parameters are mainly interactively controlled by the user, and thus of a lesser interest to our analysis.

ISOMAP, just as the previous methods, also requires the expression of neighborhoods. The main, and frequently only, exposed parameter of ISOMAP is the number of *nearest neighbors* that defines a neighborhood.

### 3.3.4 Overview comparison of algorithms

To form an impression about how the goals outlined in Sec. 3.1 are better, or less well, satisfied by LAMP, LSP, PLMP, and Pekalska, we start with an overview comparison.

Figure 3.8 shows the false neighbors, aggregated error, and most important $\phi = 5\%$ missing neighbors for the Segmentation dataset. To ease comparison, color mapping is normalized so that the same colors indicate the same absolute values in corresponding views. The aggregate error (top row) is quite similar in both absolute values and spread for all projections, *i.e.*, lower at the plot borders and higher inside, with a few dark (maximum) islands indicating the worse-placed points. Overall, thus, all studied projections are quite similar in terms of distance preservation quality. The false neighbors views (middle row) show a similar insight: Border points have few false neighbors (light colors), and the density of false neighbors increases gradually towards the projections' centers. Although local variations exist, these are quite small, meaning that all studied projections are equally good from the perspective of (not) creating false neighbors.

The missing neighbors view (bottom row) is however quite different: By looking at the size and color of the depicted bundles, we see that LSP and Pekalska have many more

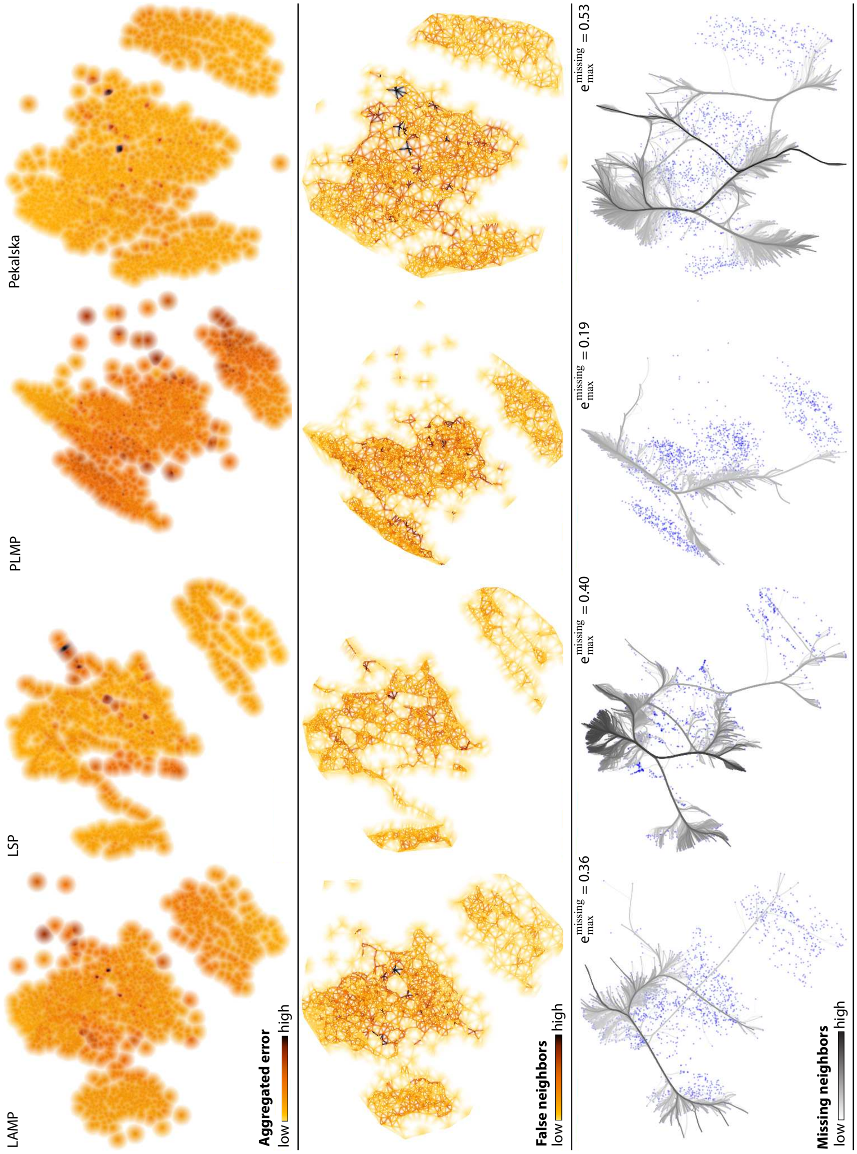


Figure 3.8: Comparison of LAMP, LSP, PLMP, and Pekalska projections for the Segmentation dataset (see Sec. 3.3.4)

.

important missing neighbors than PLMP, while LAMP has the fewest missing neighbors. In all cases, we see bundles that connect borders of the projected point-set. This confirms that all studied projections optimize placement of close points than far-away points. We also see that the missing neighbors are spread differently over the data: For LAMP, there are no bundles going to the bottom-right point cluster, showing that this cluster is indeed well separated in the projection, as it should be in relation to the $n$D data. In contrast, LSP, PLMP, and Pekalska all have bundles going to this cluster, indicating that they place these points too close to the remaining projected points.

### 3.3.5 Parameter analysis

We next refine our overview analysis by selecting two of the studied algorithms: LAMP and LSP. We next vary several of their parameters, and evaluate the resulting projections' quality with respect to this variation.

**LAMP – Different control point percentages:** Fig. 3.9 shows the results of LAMP for the Freephoto dataset with three different values for the *percentage* parameter: 10%, 30% and 50%. The error has been normalized on each view type (column in the figure).

First, we see that the final layout of the point cloud does not change drastically while varying the *percentage* parameter, only showing a 90 degree clockwise rotation for the value of 30%. While analysing the false neighbors view, we also see that, while the light brown areas are large – meaning that a moderate amount of error can be expected on the whole layout – the dark-colored spots are found nearer to the center. This suggests that LAMP positions the most problematic points in the center, surrounded by the rest of the points. By focusing on the dark spots (points with the largest false neighbor errors) throughout the parameter variation we can see that the value of the largest errors on each result remain similar – no view has many more, or much darker-colored, areas.

For the missing neighbors view, we selected a point near the upper border of the layout, marked by a cross in Figs. 3.9 (b), (e) and (h)), since missing neighbors occur mainly on the borders of the projection, as we have already observed in Section. 3.2.4. The dark spot in Fig. 3.9 (h) is where the largest error occurs over these three views. While in Fig. 3.9 (b) there are a few orange spots showing moderate error, in Fig. 3.9 (e) the error decreases considerably, and then increases again in Fig. 3.9 (h). This suggests that using about 30% of neighbors is a good value for avoiding large numbers of missing neighbors. We confirmed this hypothesis on several other datasets (not shown here for brevity). Finally, the aggregated error view shows results very similar to the false neighbors view: More problematic points (dark spots) are pushed to the center, and moderate error is found spread evenly over the entire layout. This shows that, for LAMP, most errors come from false neighbors rather than from missing neighbors.

**LSP – Different numbers of control points:** Figure 3.10 shows the same dataset (Freephoto) projected with LSP. The varying parameter is the *number of control points*. We use here the same views as in Fig. 3.9, and normalized the error in each column. By looking at the false neighbors views, we see a spatial interleaving of light-yellow and orange-brown colored areas in the projection. This contrasts with LAMP (Fig. 3.9) where

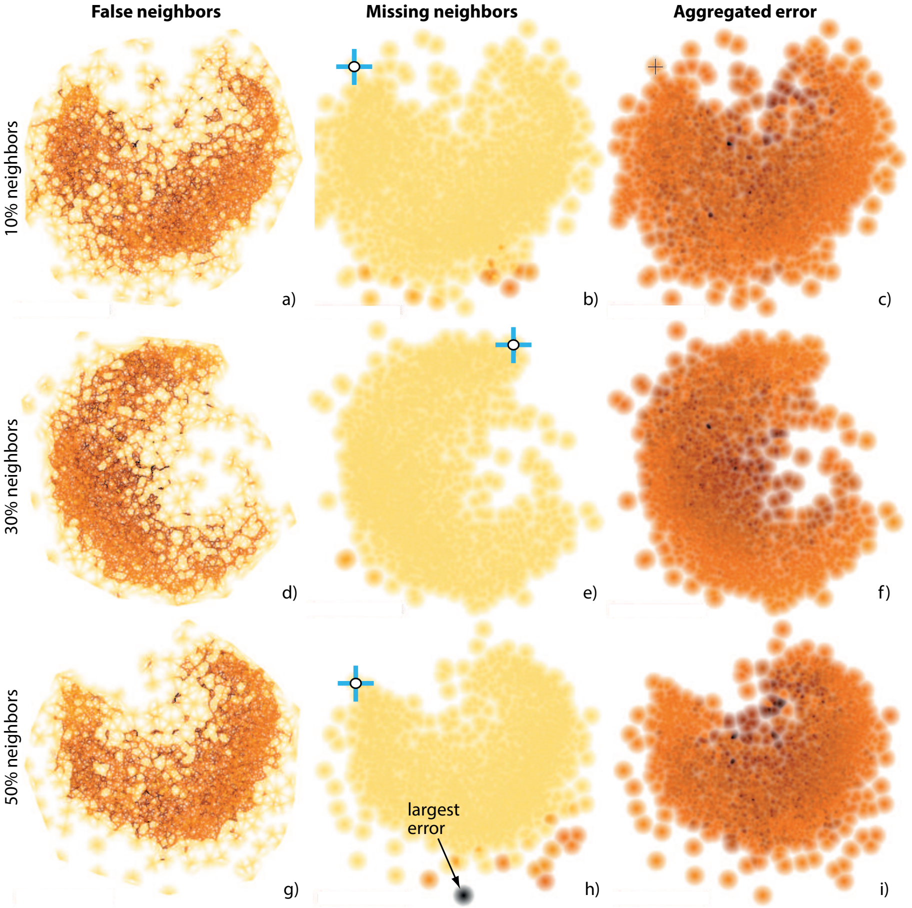


Figure 3.9: Applications – LAMP algorithm, Freephoto dataset, different neighbor *percentages* per row (see also Fig. 3.10).

the larger missing neighbor errors are consistently located away from the projection border. As the *number of control points* increases, the large error areas get more compact and closer to the projection center, but we see no increase in error severity (the amount of the orange and dark-red spots stays the same). In the missing neighbors views, the dark-colored areas in Fig. 3.10 (b) disappear largely in images (e) and (h), which means that the missing neighbors severity decreases when our control parameter increases. Comparing this with LAMP (Fig. 3.9 b,e,h), this shows that LAMP and LSP behave in opposite ways when dealing with missing neighbors. Finally, like for LAMP, the aggregate error views show the worst errors (dark spots) located in the center: The most problematic points are pushed inside by the other points which surround them, creating a mix of both false neighbors and missing neighbors. The severity of the errors, however, does not change visibly between the three parameter values.

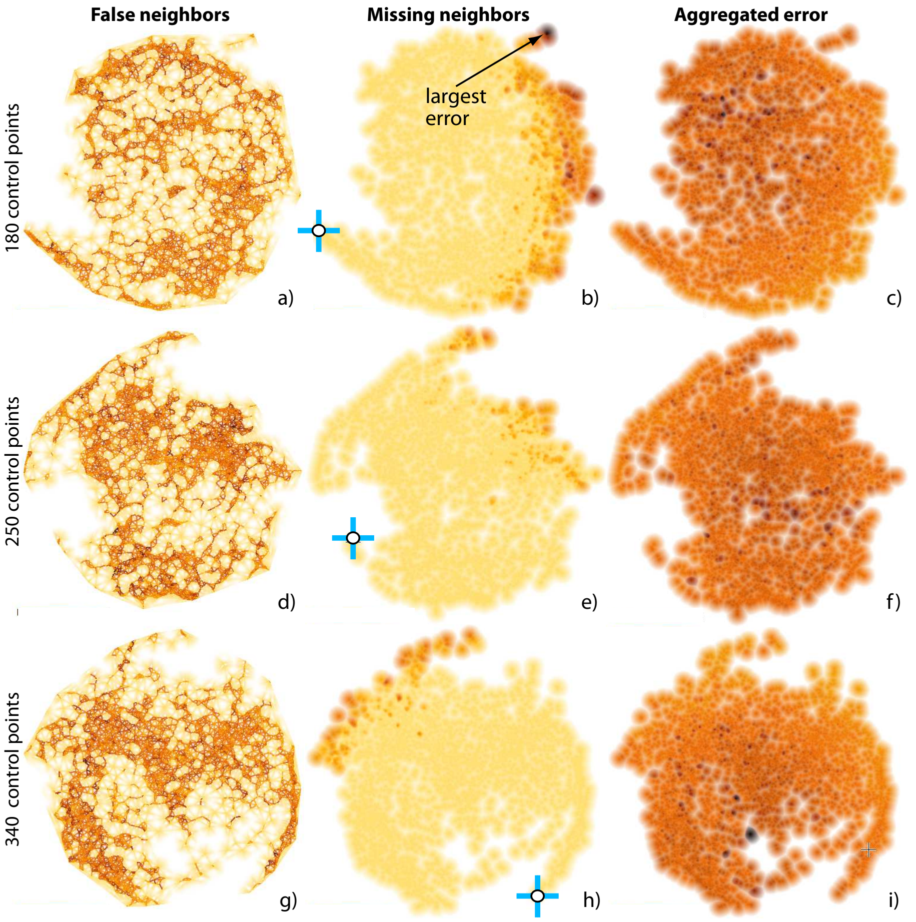
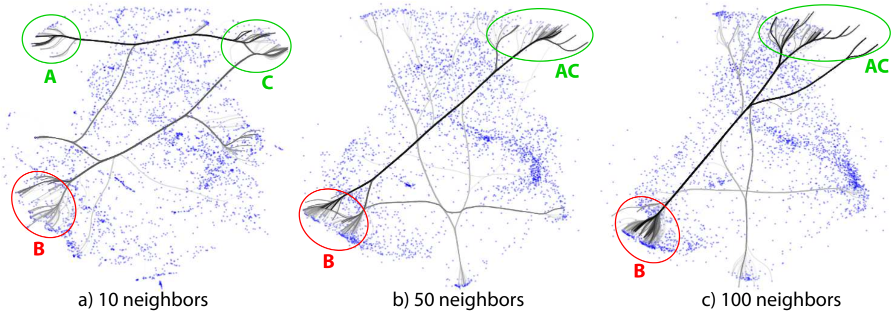


Figure 3.10: Applications – LSP technique, Freephoto dataset, different numbers of *control points* per row (compare with Fig. 3.9)

**LSP – Different numbers of neighbors:** We next examine the effect of a second parameter of LSP: *number of neighbors*. For the Freephoto dataset, we fix 250 control points and vary the number of neighbors to 10, 50 and 100. Fig. 3.11 shows the results with the missing neighbors finder view. We see that the most significant errors are initially concentrated between groups A, B and C, with C being essentially too far placed from both A and B. Increasing our parameter reduces has a positive impact on solving the missing neighbors problem between groups A and C, bringing them together into the group marked AC. The main missing neighbors are now concentrated in the relationship between groups AC and B. The 'concentration' of error given by the parameter increase is, upon further analysis, explainable by the working of LSP: Given a neighborhood $N$, LSP's Laplace technique positions all points in $N$ close to each other in the final layout.

However, the position of the neighborhoods $N_i$ *themselves* is given only by the control points, which are determined by the initial force-based layout. If this layout suboptimally places two control points $i$ and $j$ too far away from each other, then *all* points within the neighborhoods $N_i$ and $N_j$ end up being too far away from each other. Hence, as the neighborhood size increases, the likelihood to see fewer thick high-error bundles increases. This insight we found is interesting since it was not reported in the LSP literature so far, and it can be explained (once we are aware of it) by the algorithmics of LSP.



a) 10 neighbors b) 50 neighbors c) 100 neighbors

Figure 3.11: Applications – LSP technique, Freephoto dataset, different numbers of *neighbors*. Bundles show most important missing neighbors.

**LAMP – Different datasets:** We next analyse the LAMP technique applied to three different datasets: Corel (1000 elements), Freephoto (3462 elements), and Sourceforge (6773 elements). The varying parameter is now the input *dataset* itself. The aim is to see whether (and how) errors are affected by the nature of the input data, *e.g.* distribution of similarity, number of dimensions, and number of points. Figure 3.12 top row shows the false neighbors views. We see here that, while for the first two datasets the behavior of false neighbors is similar to earlier results, for the largest dataset (Sourceforge) there are much fewer false neighbors. These are located close to the intersection area of the two apparent groups in the image, and on the borders of these groups. This, and the low errors (light colors) inside the groups may indicate that both groups have a high degree of cohesion between their inner elements. The large errors on close to the intersection areas and borders can indicate elements that could be in either group, respectively very different from all other elements. Figure 3.12 (a) shows a similar pattern: Most false neighbors are located at the 'star' shape's center, while the arms of the start contain elements that are more cohesive. This may indicate that the dataset contains a number of cohesive groups equal to the number of start arms, and elements in the center belong equally to all groups.

While analysing the missing neighbors for several points selected on the periphery of the projections, we see that the errors are smaller for Figs. 3.12 (d) and (e), and considerably larger for Fig. 3.12 (f). For the last image, we selected a point close to the intersection area of the perceived groups. Image (f) shows that this point is *equally* too far placed from most points in *both* perceived clusters. The size and speed of increase of

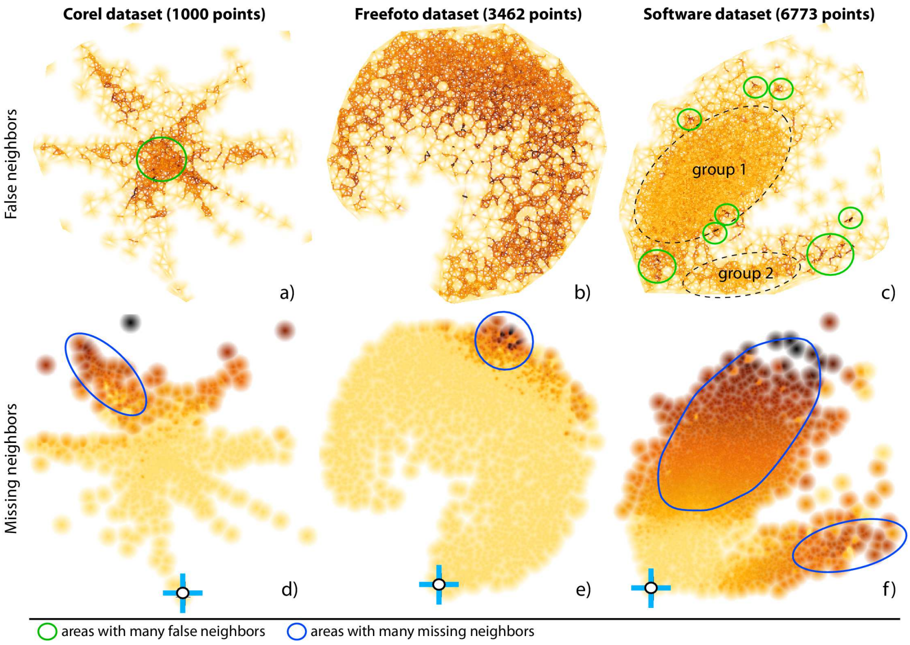


Figure 3.12: Applications – One algorithm (LAMP), different datasets. Top row: false neighbors. Bottom row: missing neighbors.

the error (as we get further from this point in the projection space) strongly suggests that the selected point belongs stronger to both perceived groups than the projection indicates. This strengthens our initial hypothesis that the area separating the two groups belongs equally to these groups.

**ISOMAP – Different numbers of neighbors:** To illustrate a different type of analysis made possible by our work, Fig. 3.13 shows the effect of changing the number of *neighbors* in ISOMAP on missing group members. Our group $\Gamma$ of interest, shown first on Fig. 3.13 (a), is highlighted in images (b-d) by a shaded cushion. Besides the fact that $\Gamma$ moves from the left of the projection to the right, images (b-d) show how its missing members behave as we change our parameter. At first, in Fig. 3.13 (b), we see that the most important missing neighbors are found in two other areas $A_1$ and $A_2$ on the far side of the layout. We also notice many black edges, which means that the points in $A_1$ and $A_2$ are indeed too far away from all points in the selected group. The relatively large fan-out of the bundles show that the group misses many members, and these are scattered widely over the projection. As the parameter increases, we see in image (c) that the missing members spread out even more, but the severity of the errors decreases (as shown by the lighter colors of $e_\Gamma^{missing}$ background. The inner fanning of the edges, inside $\Gamma$, is still large, which shows that many group members miss neighbors. Finally, in Fig. 3.13 (d), issues decrease significantly: We see thinner bundles, which imply less error; the bundle fanning inside $\Gamma$ is relatively small, meaning that most of

Γ's points do not miss neighbors; and the fan-out of the bundles is smaller, showing that the missing group members are now more concentrated than for the first two parameter values. This leads to the conclusion that, for the analysed group, the increase of the number of neighbors parameter has a positive impact on the final projection quality.

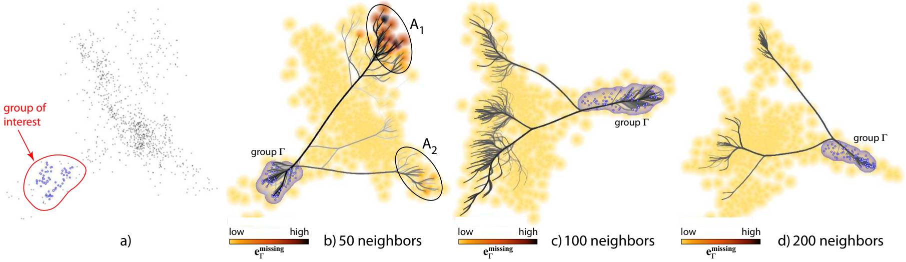


Figure 3.13: Applications – ISOMAP projection, finding missing group members for different numbers of *neighbors*.

**LSP – Different numbers of iterations:** The final analysis we present compares two different LSP projections of the same dataset (News), computed using values of 50, respectively 100 for the *number of iterations* parameter of the control-point force-directed placement.

Figures 3.14 (a) and (b) show the two LSP projections. In each of them, several high-density groups are visible. These are strongly related news feeds, *i.e.*, which likely share the same topic (see Sec. 3.3.1). However, without extra help, we cannot *relate* the two projections, *e.g.*, find out (a) if points significantly change places due to the parameter change; (b) which groups in one projection map to groups in the other projection; and (c) whether points in a group in one projection are also grouped in the second projection.

To answer question (a), we use the projection comparison view (Sec. 3.2.7). The result (Fig. 3.14 (c)) shows that there are many large point shifts; the bundle criss-crossing also shows that groups change places in the projection. This is a first indication that LSP is not visually stable with respect to its number of iterations parameter. Next, we manually select three of the most apparent point groups in one projection, shown in Fig. 3.14 (a) by the shaded cushions $A$,$B$,$C$. We examine these in turn. In Fig. 3.14 (d), we show how points in group $A$ shifted, in the second projection, to a group $A_1$. Virtually all bundled edges exiting $A$ end in $A_1$, so the parameter change preserves the cohesion of group $A$ (though, not its position in the layout). The same occurs for group $B$ (Fig. 3.14 (e)). However, the parameter change spreads $B$ more than $A$ – in image (e), we see that $B$ maps to three groups, $B_1..B_3$. These visualizations thus answer question (b). Group $C$ behaves differently (Fig. 3.14 (f)): This group is split into two smaller groups $C_1$ and $C_2$ when we change our parameter. For question (c), thus, the answer is partially negative: not all groups are preserved in terms of spatial coherence upon parameter change.

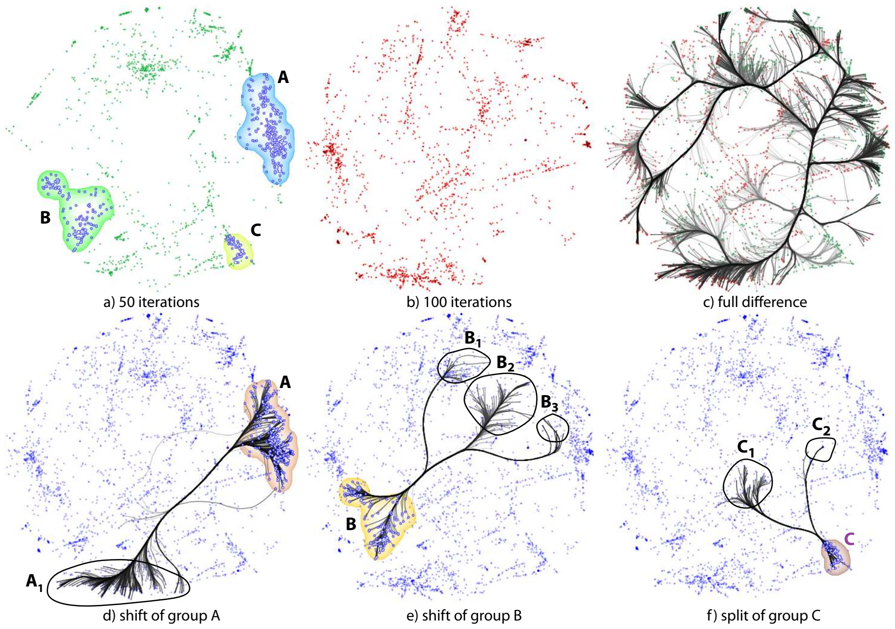


Figure 3.14: Applications – Shift between two LSP projections, for different numbers of force-directed *iterations*.

## 3.4 Discussion

We have implemented our visualization techniques in C++ using OpenGL 1.1, and tested them on Linux, Windows, and Mac OSX. Below we discuss several aspects of our techniques.

**Computational scalability:** For Delaunay triangulation and nearest-neighbor searches, we use the Triangle [167] and ANN [6] libraries. Both can handle over 100K points in subsecond time on a commodity PC. Further, we accelerate imaging operations using GPU techniques. For 2D distance transforms, we use the pixel-accurate Euclidean distance transform algorithm and GPU implementation proposed in [28]. On an Nvidia GT 330M, this allows us to compute shaded cushions and perform our Shepard interpolation at interactive frame rates for views of $1024^2$ pixels. For edge bundling, we implemented KDEEB [85] fully on Nvidia's CUDA platform. This yields a speed-up of over 30 times (on average) as compared to the C# implementation in [85] and allows bundling graphs of tens of thousands of edges in roughly one second. All in all, we achieve interactive querying and rendering of our views for projections up to 10K points.

**Visual scalability:** Our image-based approaches scale well to thousands of data points or more, even when little screen space is available. Moreover, all our techniques have a multiscale aspect: The parameters $\alpha$ and $\beta$ (Eqns. 3.4, 3.5) effectively control the visual *scale* at which we want to see false neighbors, missing neighbors, and the aggregate error. Increasing these values eliminates spatial outliers smaller than a given size, thereby emphasizing only coarse-scale patterns (see *e.g.* Fig. 3.1). The bundled views (Sec. 3.2.5) also naturally scales to large datasets given the inherent property of bundled edge layouts to emphasize coarse-scale connectivity patterns.

**Genericity:** Our visualizations are applicable to any DR algorithm, as long as one can compute an error distance matrix encoding how much 2D distances deviate from their $n$D counterparts (Eqn. 2.2). No internal knowledge of, or access to, the DR algorithms is needed – these can be employed as black boxes. This allows us to easily compare widely different DR algorithms, *e.g.* based on representatives, based on distance matrices, or based on direct use of the $n$D coordinates.

**Ease of use:** Our views are controlled by three parameters: $\alpha$ sets the scale of the visual outliers we want to show; $\beta$ sets the radius around a point in which we want to display information, *i.e.*, controls the degree of space-filling of the resulting images; $\phi$ sets the percentage of most important missing neighbors we want to show. These parameters, as well as the interaction for selecting point groups (Sec. 3.2.6) are freely controllable by users by means of sliders and point-and-click operations.

**Comparison:** Similarly to Van der Maaten *et al.* [193], we use multiple views showing the same data points to explain a projection, *e.g.*, the false neighbors, missing neighbors view, missing neighbors finder, and group-related maps. However, the multiple maps in [193] are used to actually convey the projection, so the same point can have different locations and/or weights in different maps. In contrast, we use multiple views to convey different quality metrics atop of the same 2D projection, but keep the position of all observations the same in all these views. This simplifies the user's task of correlating these multiple views, based on spatial positions. Similar to Aupetit [8], our error metrics encode discrepancies in distances in $\mathbb{R}^n$ *vs* $\mathbb{R}^2$. However, our error metrics are different. More importantly, our visualizations are different: Our false neighbors view does not show (a) spurious Voronoi cell edges far away from data points or (b) cell subdivision edges whose locations does not convey any information, since we (a) use distance-based blending and (b) continuous rather than constant per-cell interpolation (Sec. 3.2.3). Secondly, our missing neighbors finder (Sec. 3.2.5) can show one-to-many and many-to-many error relationships, whereas all other methods are constrained to one-to-one relationships. Finally, we can show errors at group level, whereas the other studied techniques confine themselves to showing errors at point level only.

Our projection comparison view (Sec. 3.2.7) is technically related to the method of Turkay *et al.*, which connects two 2D scatterplots to each other by lines linking their corresponding points [189]. However, Turkay *et al.* stress that line correspondences only work for a *small* number of points. In contrast, we use bundles to (a) show up to thousands of correspondences, and coloring and blending to encode correspondence importance.

**Findings:** It can be argued that our results are limited, as we did not decide, using our method, which of the studied DR algorithms are best. However, this was not the aim of our work. Rather, our goal was to present a set of visual techniques that help analyse the effect of parameters on projection quality for several DR techniques of interest. Deciding whether a certain degree of quality, *e.g.* in terms of false neighbors, missing neighbors, grouping problems, or projection stability is a highly context, dataset, and application-dependent task. Having such a context, our tools can be then used to assess (a) which are the quality problems, (b) how parameter settings affect them, and (c) whether these problems are acceptable for the task at hand. The same observation applies to the datasets used here. Our analyses involving these should be seen purely as test cases for assessing the quality problems of DR projections, and not as findings that affect the underlying problems captured by these datasets.

**Distance *vs* neighborhood preservation:** It can be argued that our error metrics, and corresponding visualizations, measure and respectively present a mix between the preservation of distances and neighborhoods: The aggregate error view is a 'pure' distance-related metric, very close to the well-known aggregated stress metric. The false neighbors view encodes distance errors into colors, but only for the Delaunay neighbors of each projected point. The missing neighbors view is also a pure distance-related metric. The missing neighbors finder highlights, indeed, the most important missing neighbors of a projected point, but uses a distance metric to define the notion of missing neighbors. The group analysis views lift the above interpretations at the level of a user-chosen group of projected points.

Obviously, distance preservation and neighborhood preservation are related notions: Perfect distance preservation implies perfect neighborhood preservation, as nearest neighbors are defined by distances. However, the converse is not true – neighborhoods are defined in terms of the order of points as given by their sorted set of distances, and not by the absolute value of distances. As such, very high distance-preservation errors are quite likely to cause also significant errors in neighborhood preservation, but the two types of errors are not reducible to each other. As all our error metrics proposed in this chapter are essentially distance-based, we refer to them as distance-preservation errors. In contrast, errors that use sets of nearest neighbors in their formulation, rather than distances, will be called neighborhood-preservation errors, and will be analysed separately in Chapter 4.

**Limitations:** A few limitations can be identified for the techniques presented in this chapter. The error metric described by Equation 3.1 can be quite sensitive to outliers. If there is an outlier in the original space ($D^n$) then $\max_{i,j} d^n(\mathbf{p}_i, \mathbf{p}_j)$ will be much larger than most other pairwise distances $d^n$. Hence, after naive linear normalization, the distances between all non-outlier points will be squeezed into a small portion of the output (normalized) space. Unless this outlier behavior is very well-represented in the projection ($D^m$), this will make the rightmost term of Equation 3.1 be quite different than the leftmost term even when they should be similar, so well-positioned points may still show errors. Considering also the phenomenon of 'curse of dimensionality', it can be very hard for projections to represent outliers well without crowding non-outlier points in a very small area in the output space. The neighborhood-preservation error metrics proposed in the next chapter deal with this by considering, for each point $i$, only the ranks of neighbors of $i$, i.e., their discrete position in the sorted list of nearest-neighbors of $i$, and not their actual distances.

Regarding our visualizations, as outlined by the examples, they can show (a) which projection areas suffer from low quality; and (b) how two projections differ in terms of neighborhood preservation. However, we cannot directly explain (c) *why* a certain DR algorithm decided to place a certain point in some position; and (d) how the user should *tune* (if possible) the algorithm's parameters to avoid errors in a given area. In other words, we can explain the function $f : P$ (Eqn. 2.1) and its first derivatives over $P$, but not the inverse $f^{-1}$. This is a much more challenging task – currently not solved by any technique we know of. Further explaining such second-order effects to help users locally fine-tune a projection is subject to future work. Secondly, the parameter space $P$ of some DR algorithms can be high-dimensional. So far, we can only analyse the variation of one or two parameters at a time. Extending this to several parameters is a second challenging next topic.

## 3.5 Conclusions

We have presented a set of visualization methods for the analysis of the quality of dimensionality-reduction (DR) algorithms in terms of their ability to preserve distances. We generically model such algorithms as functions from $n$D to 2D, parameterized in terms of the various settings of the respective projection algorithm. Next, we classify distance-related projection errors into false neighbors, missing neighbors, and aggregated projection error at both individual point and point-group level, and propose metrics to quantify these errors. We next propose several dense-pixel, visually scalable, techniques such as multi-scale scattered point interpolation and bundled edges to display out error metrics in ways that are visually and computationally scalable to large datasets and also work in a multiscale mode. We demonstrate our techniques by analysing the parameters of five state-of-the-art DR techniques.

In contrast to existing assessments of DR projections by aggregate figures, that can only infer overall precision, we offer more local tools to examine how neighborhoods and groups are mapped in the final projection. The usage of our techniques is simple and, most importantly, allows users of DR techniques to study their quality without needing to understand complex internal processes or the exact role of each parameter in the projections.

As noted in the introduction of this chapter, the techniques presented here only deal with distance errors in projections. While these are, naturally, important to quantify and see, they are not the only sources of problems in interpreting projections. As such, in Chapter 4, we will detail the issue of quantifying and visualizing neighborhood-preservation errors, the second most-important source of projection interpretation errors identified in Chapter 2. Separately, Chapter 5 will show how projection errors can be computed and used to assess distance preservation for 3D projections, and also compare the quality of 2D and 3D projections generated by the same projection technique.

After identifying the nature, distribution, and magnitude of errors in a given projection, a natural question for users is to assess which points get affected by such errors. This requires explaining groups of points in a projection in terms of the underlying high-dimensional variables. A related task of interest is to understand why errors appear for a given subset of points of a given dataset projected by a given algorithm. Both above tasks can be addressed by depicting the most important high-dimensional variables that cause a projection to place points close to, or far away from, each other. Techniques that address these tasks will be discussed in Chapter 6.

## Contributions

The text of this chapter is based on the article "Visual analysis of dimensionality reduction quality for parameterized projections" (R. Martins, D. Coimbra, R. Minghim, A. Telea), *Computers & Graphics*, vol. 41, pp 26-42, 2014. The two first co-authors have had equal major contributions to this publications, and should be seen as joint first authors. Specific contributions of R. Martins involve: the proposal of dense image-based techniques to encode and visualize various forms of projection errors, based on smooth interpolation, and the design of the various error metrics (Sec. 3.2.2 and following); the adaptation and usage of edge bundles to visualize various types of errors (Secs. 3.2.5 and 3.2.6); and the selection, usage, and interpretation of projections constructed by the PLMP, Pekalska, and ISOMAP techniques (Sec. 3.3.2 and following).

## 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:

- $\textbf{missing neighbors} = \{\mathbf{p}_j \in \nu_k^n(i) \wedge \mathbf{q}_j \notin \nu_k^2(i)\}$
  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 $\mathbf{q}_i$.
- $\textbf{false neighbors} = \{\mathbf{p}_j \notin \nu_k^n(i) \wedge \mathbf{q}_j \in \nu_k^2(i)\}$
  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 $D^n$.
- $\textbf{true neighbors} = \{\mathbf{p}_j \in \nu_k^n(i) \wedge \mathbf{q}_j \in \nu_k^2(i)\}$
  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 $k$.

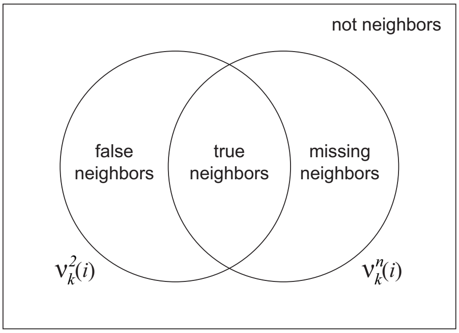


Figure 4.1: The four types of points that can be derived from the two $k$-neighborhoods $\nu_k^n(i)$ and $\nu_k^2(i)$ of a point $i$ when analysing neighborhood preservation.

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

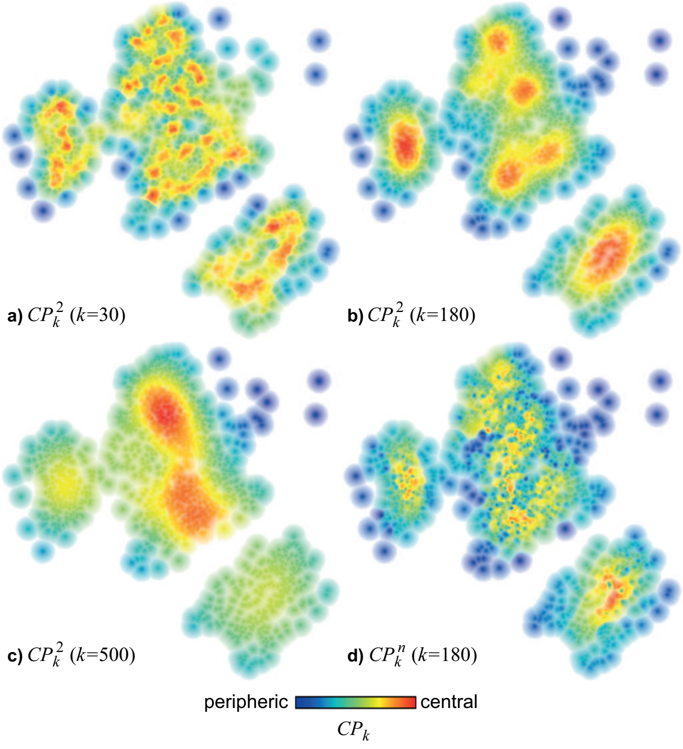


Figure 4.2: *Centrality preservation* view, *segmentation* dataset, LAMP projection. (a-c) Centrality $CP_k^2$, for three neighborhood sizes $k$. (d) Centrality $CP_k^n$, for $k = 180$ neighbors.

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.

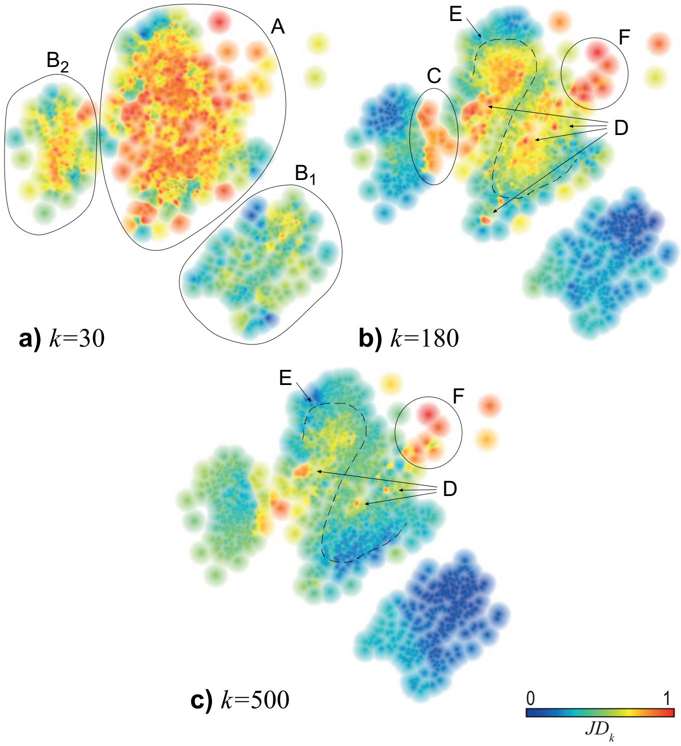
Fig. 4.3 shows three set-difference views for the same $k$ values as in Fig. 4.2, for ease of comparison. According to the interpretation of $JD_k$, high values (warm colors) show poor neighborhood preservation while low values (cold colors) show areas where neighbors are well preserved. With these new 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,



**a)** $k$=30 **b)** $k$=180

**c)** $k$=500

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 $\rho_i^2$ 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)

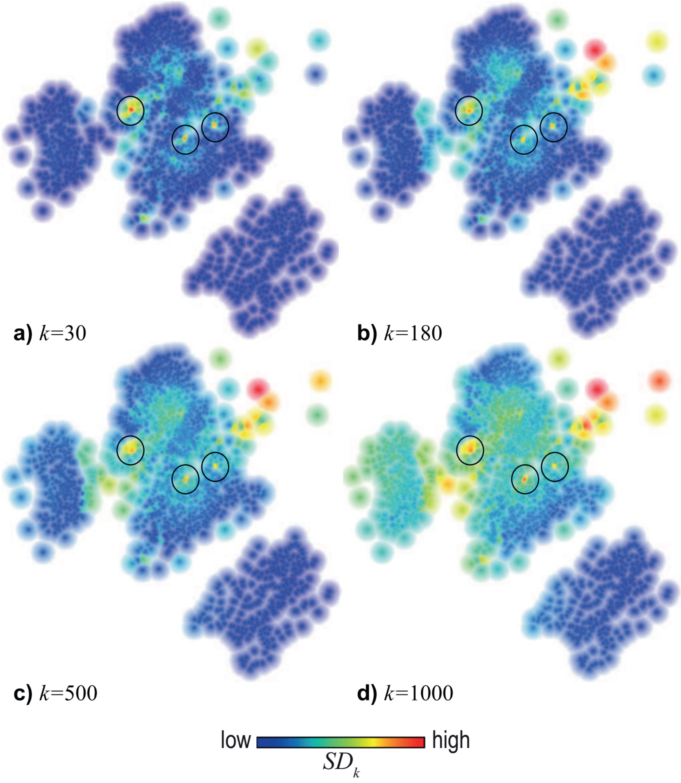


Figure 4.4: *Sequence difference* view, *segmentation* dataset, LAMP projection, for four increasing scales ($k$ values).

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.

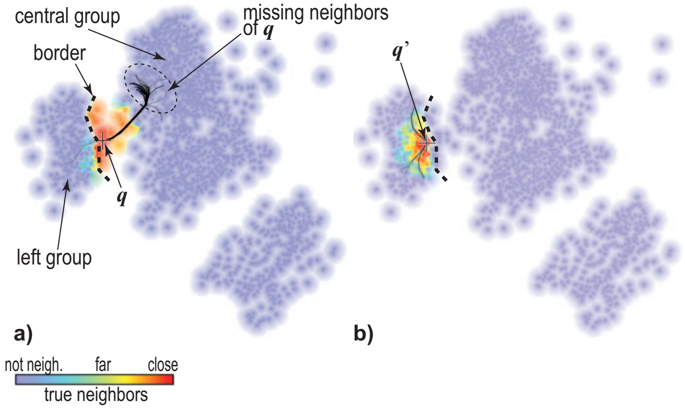


Figure 4.5: Local analysis of the connection between the left and central groups. The visual border, seen also in Figs. 4.3b and 4.4, is marked by a dotted curve.

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.

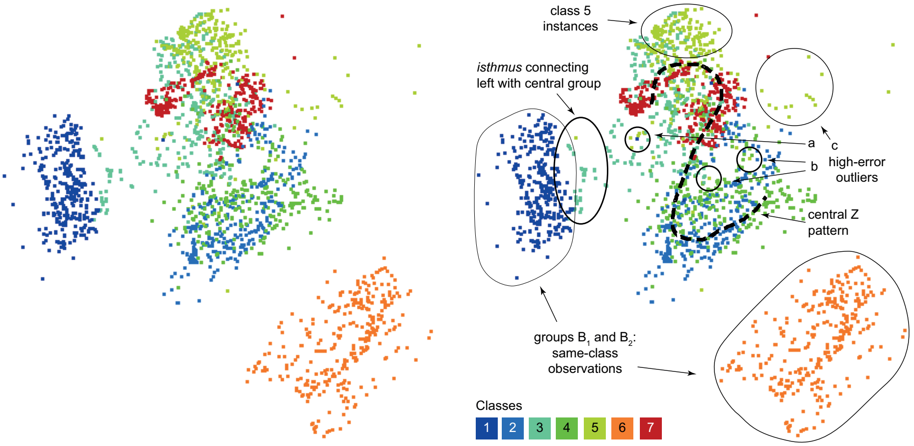


Figure 4.6: Original classification (*ground truth*) for the *segmentation* dataset.

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.

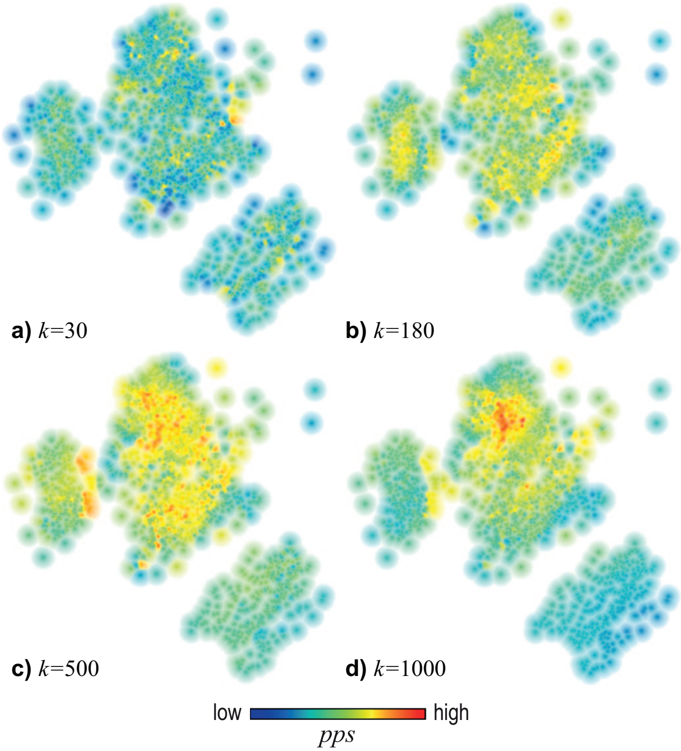


Figure 4.7: Projection precision score (*pps*) [159], *segmentation* dataset, LAMP projection, for four scales ($k$ values).

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.

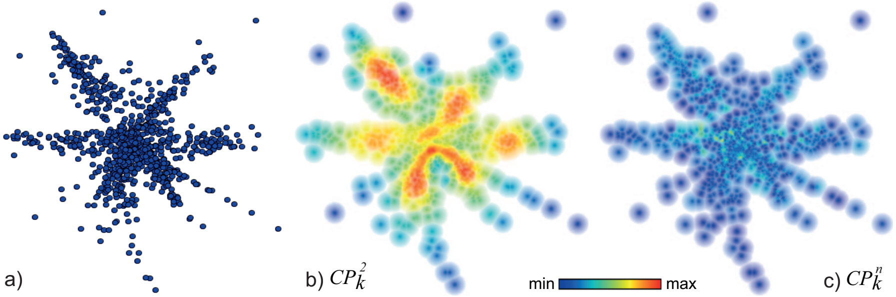


Figure 4.8: Corel dataset, LAMP projection, $k = 75$ neighbors. (a) Projection without error metrics. (b) Projection colored by $CP_k^2$. (c) Projection colored by $CP_k^n$.

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.

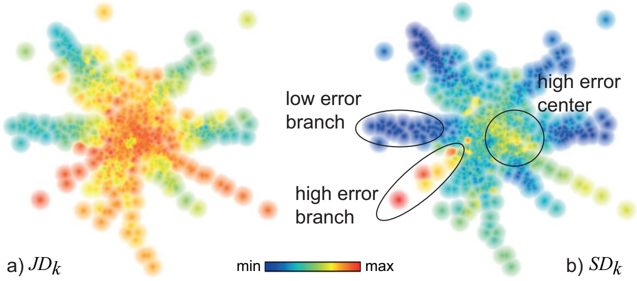


Figure 4.9: Corel dataset, LAMP projection, $k = 75$ neighbors. (a) Set difference view. (b) Sequence difference view.

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

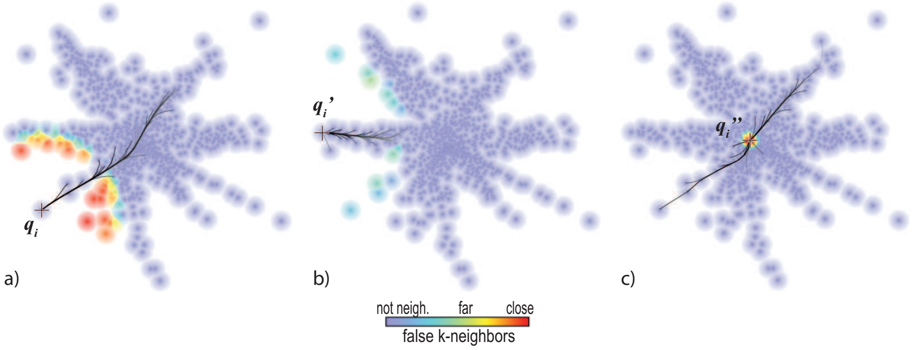


Figure 4.10: Local neighborhood analysis for the Corel dataset, LAMP projection, $k = 75$ neighbors.

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

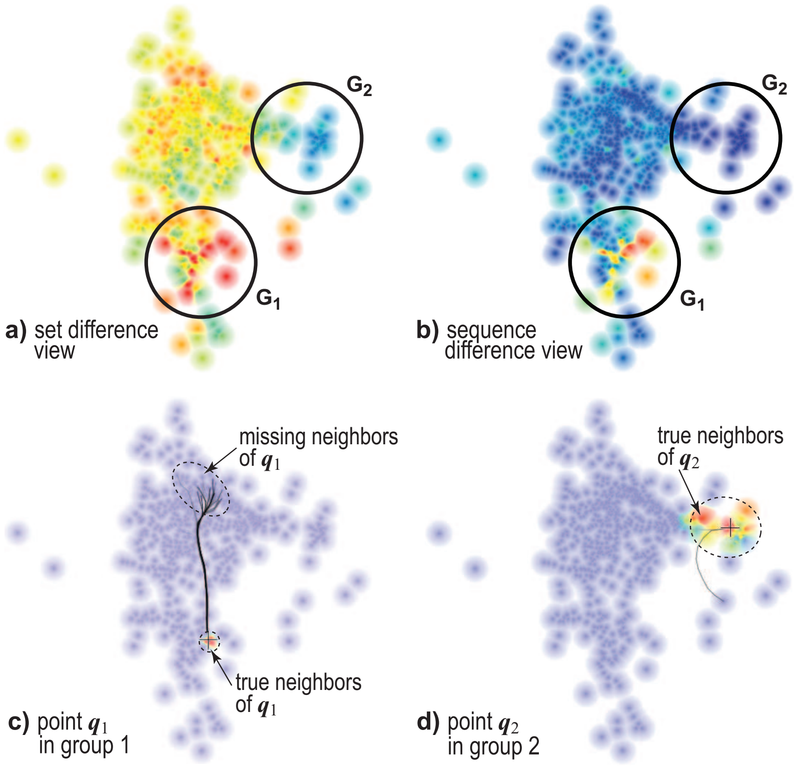


Figure 4.11: Neighborhood-preservation analysis for the *Github* dataset, LSP projection, $k = 72$ neighbors.

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 exploratory 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.

Chapter 5

# Explaining 3D Multidimensional Projections

The previous two chapters have presented several examples showing how multidimensional projections can help in understanding the structure of high-dimensional datasets by creating two-dimensional (2D) representations thereof. Such representations can, next, be used to reason about data features such as groups of observations, outliers, and trends. Separately, we have shown how to augment 2D projections to show where errors occur in the projection process with respect to the lack of preservation of distances and/or neighborhoods of observations. Showing such errors is essential to gauge the degree of trust one can assign to the discovered structures in the 2D projection, in terms of knowing how much these structures reflect actual structures present in the high-dimensional data or, alternatively, whether these structures are projection artifacts.

Our results shown in Chapters 3 and 4 show that projection errors, in terms of distance and/or neighborhood preservation, occur up to various degrees for all the studied projection techniques and datasets that we have analysed. As outlined in the respective chapters, this result is not surprising, and it is caused by the large difference in dimensionality between the input high-dimensional space and the output 2D space where the projection resides.

Given the above, three-dimensional (3D) projections are an interesting alternative to two-dimensional projections in terms of reducing projection errors. Indeed, the extra dimension offered by 3D projection diminishes the dimensionality difference outlined above and, as such, reduces projection errors [97, 139, 148]. Additionally, for state-of-the-art projection techniques, the computational and end-user effort required to generate 3D projections is practically identical to the one required to generate corresponding 2D projections. As such, 3D projections represent an interesting alternative to 2D projections when exploring high-dimensional datasets.

However, multidimensional 3D projections come with additional challenges, as follows:

- *Accuracy:* While, in general, 3D projections have a lower projection error as compared to their 2D counterparts, the assessment of such projection errors has been done so far using mainly aggregated projection errors [97, 139, 148]. For a fair comparison of 2D and 3D projections, a more fine-grained error assessment is required, in line with the techniques we proposed for 2D projection error assessment in Chapters 3 and 4.
- *Usability:* 3D projections output a 3D point cloud, which is next visualized usually as a scatterplot. However, in contrast to classical 3D scatterplots whose axes have clear meanings (they map data variables one-to-one), the axes of a 3D projection do not carry any particular meaning. As such, interpreting such scatterplots is challenging in an absolute sense [50], but also in a relative sense, if we compare them with the easier to interpret 2D scatterplots created by projections [128,

> 205, 163]. As users rotate the 3D scatterplot to find a suitable viewpoint, several questions arise, such as: How much of the original data structure has the projection preserved? What is the meaning of the 3D directions along which scatterplot points are spread, in terms of original variables? How can we see values of these variables in the scatterplot? What are good viewpoints to look at the scatterplot from, given a set of questions on these variables?

To support the claim that 3D projections are effective tools for exploring multidimensional datasets, we have to study both above points in detail. Indeed, for a 3D projection to be indeed *effective*, it has to (1) provide measurable added-value in terms of reduced errors, as compared, again, to a 2D projection; and (2) be easy to use, at least up to a similar level as compared to a 2D projection.

In this chapter, we address the above two aspects concerning the explanation and error-assessment of 3D dimensionality reduction (DR) projections, as follows. First, to address point (1) outlined above, we extend our error metrics proposed in Chapter 3 to handle 3D projections (Sec. 5.1.1). Separately, we use these error metrics to compare 3D projections with their 2D counterparts, for a given projection technique and dataset. This provides a way to compare the added-value of 3D projections *vs* their 2D counterparts in a quantitative and detailed way, and thereby supports the general argument made for the added-value of 3D projections. Secondly, we propose a set of interactive explanatory visualization techniques to help users answer the questions listed under point (2) above for 3D DR projections (Sec. 5.1.2 and following). Our techniques work as add-ons to any DR technique, *i.e.*, do not depend on technical aspects of the DR algorithm being used. We keep their visual design simple, so that learning to use them requires limited effort. While we also use interaction to explain a projection, like [51, 210, 134], our focus is to explain projection-space distances in terms of the original $n$-D *variables*, rather than showing similarities of projected points with a user-selected set of variable values or extracting higher-level semantics from variable values. As such, we do not modify our projection, as we consider it to be our 'ground truth', and also give a key role to the $n$-D variables in our explanation. We integrate our techniques with classical 3D scatterplot views, so that they can be readily used to assist typical projection-exploration scenarios, or in other words, *explain* the projection. We illustrate our visualization techniques by applying them to several data exploration scenarios involving real-world multidimensional datasets and a set of recent DR projection algorithms.

The structure of this chapter is as follows. Section 5.1 introduces our explanatory visualizations for 3D projections, both in terms of understanding the projection errors and in terms of understanding which variables are best explained by a given viewpoint of the corresponding 3D projection, via a simple dataset. Section 5.2 illustrate how our visualizations can answer several questions on 3D scatterplots created by several DR techniques from real-world datasets, thereby showing how these techniques augment the usability of 3D projections. Section 5.3 discusses our techniques. Finally, Section 5.4 concludes this chapter.

## 5.1 Explanatory Visualizations

To illustrate our proposal next, we choose a simple sample dataset. This dataset has 2814 points, each having 9 attributes (dataset $ALL$ in [136]). The points represent scientific papers. The dimensions were created by using stemming and stop-word elimination on the text of the scientific abstracts of the respective papers, followed by calculating the term-frequency-inverse-document-frequency count [154], well known in text mining.

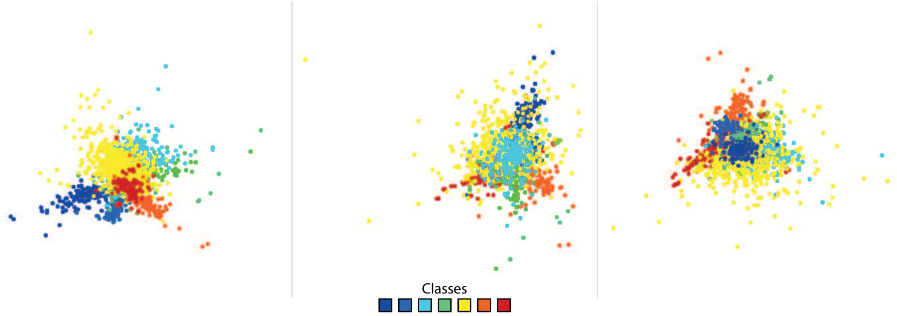


Figure 5.1: Three different viewpoints of the 3D LAMP projection of the *ALL* dataset [136].

Our first visual exploration of this dataset, shown in Fig. 5.1, depicts the data using a 3D projection, computed by the LAMP technique LAMP [95]. Projected points are color-coded by an additional categorical class attribute – not used in the projection – using the categorical colormap indicated in the figure. This attribute describes the topic of the respective documents, and was assigned manually to the dataset, after inspection of the abstracts. Three 3D viewpoints of this visualization are shown in Fig. 5.1.

Looking at the images in Fig. 5.1, the only (relatively) salient aspect one can find, is the presence of a number of relatively compact same-color points in the scatterplot. Assuming the LAMP projection indeed preserves distances, this would indicate that documents whose topics (as found by humans) are similar, also have similar high-dimensional attributes. If so, this would be an interesting finding, further justifying the use of the respective attributes for *e.g.* automatic topic mining or similar tasks [136].

However, the insight provided by our 3D projection stops here. Besides seeing the above-mentioned color-coherent point groups, there is little other insight we can directly get from the image. For instance, we cannot see how the nine dimensions of the dataset correlate (or not) along the projected points, or along specific point clusters; we do not know which specific variables (dimensions) or value-ranges thereof define a specific cluster; we do not know how are the distances and/or neighborhoods of the original space preserved, by this specific technique, in three dimensions (thus, how much we should trust what the 3D projection shows); and finally, we have no guidance in choosing suitable viewpoints that best support a given question concerning the data.

As such, this 'raw' 3D projection is not very useful: It tells us *that* the nine attributes extracted from the document dataset seem to correlate relatively well with the ground truth captured by the class attribute, which is a desirable situation for *e.g.* further building automatic classification systems for such documents. However, this image does not tell us *how* attributes correlate with document classes or *how well* the final layout represents the original data. The remainder of this chapter presents methods and techniques to address the above two points.

### 5.1.1 Accuracy of 3D projections

We start our exploration of 3D projections with an analysis of how the errors are distributed in such projections. In turn, this will help us to gauge the added value of 3D representations over 2D ones in terms of the accuracy of the representation of the multidimensional data space. To do the above error assessmebt, we adapt one of the techniques we introduced in Chapter 3: the aggregated distance-based error. While this method was initially used only for 2D scatterplots created by 2D projections, it can be readily applied to 3D with minimal adaptation. Recall that, as defined in Section 3.2.1, the aggregate normalized projection error $e_i^m \in [0,1]$ is

$$e_i^m = \sum_{j \neq i} \left| \frac{d^m(\mathbf{q}_i, \mathbf{q}_j)}{\max_{i,j} d^m(\mathbf{q}_i, \mathbf{q}_j)} - \frac{d^n(\mathbf{p}_i, \mathbf{p}_j)}{\max_{i,j} d^n(\mathbf{p}_i, \mathbf{p}_j)} \right|. \tag{5.1}$$

Here, $d^n$, $d^m$, $\mathbf{p}$, and $\mathbf{q}$ are identical to the respective terms introduced earlier in Eqn. 2.2. Note also that $e_i^m$ is essentially a different way to scale the aggregated normalized projection error $e_i^{aggr}$ (Eqn. 3.2) introduced in Sec. 3.2.2. The error $e_i^m$, $m \in \{2,3\}$, briefly put, measures how well the distances between a projected point $i$ in $m$D, to all other points $j \neq i$, reflect the same distance-pairs between points $(i,j)$ in the original $n$-dimensional space. For details, we refer to Section 3.2.2, where we have introduced the aggregate normalized projection error.

To compute and compare the error of a 3D projection with the similar error of a 2D projection, we propose three different formulations, as follows.

**2D projection error:** For the studied $ALL$ dataset, we start by computing its projection error as given by projecting the 9-dimensional dataset to two dimensions, using the LAMP technique. Figure 5.2a shows this error, which was computed in the exact same way as described in Chapter 5.1 – that is, by comparing the original 9-dimensional inter-point distances with the distances between the same points in the 2D projection space. The result is a layout with a dense grouping of points in the center and some outstretching spearated clusters that spread away towards the borders of the image, in different directions. Distances are well-preserved in the center of the projection, despite the cluttering, with some high-error outliers on the periphery and a tendency of mid-level errors towards the tips of the outstretching clusters.

**3D projection error:** A second way to compute the projection error is to use a 3D projection rather than a 2D one. In this case, the final projection space has $m = 3$ dimensions rather than $m = 2$, as used earlier in Fig. 5.2a. Figures 5.2b–c show the error $e_i^m$ computed for $m = 3$ dimensions, by directly applying Eqn. 5.1 with $m = 3$ instead of $m = 2$. Given that the error formulation is view-independent, its values will be the *same* (for the same 3D points) in any different viewpoints for the same projection.

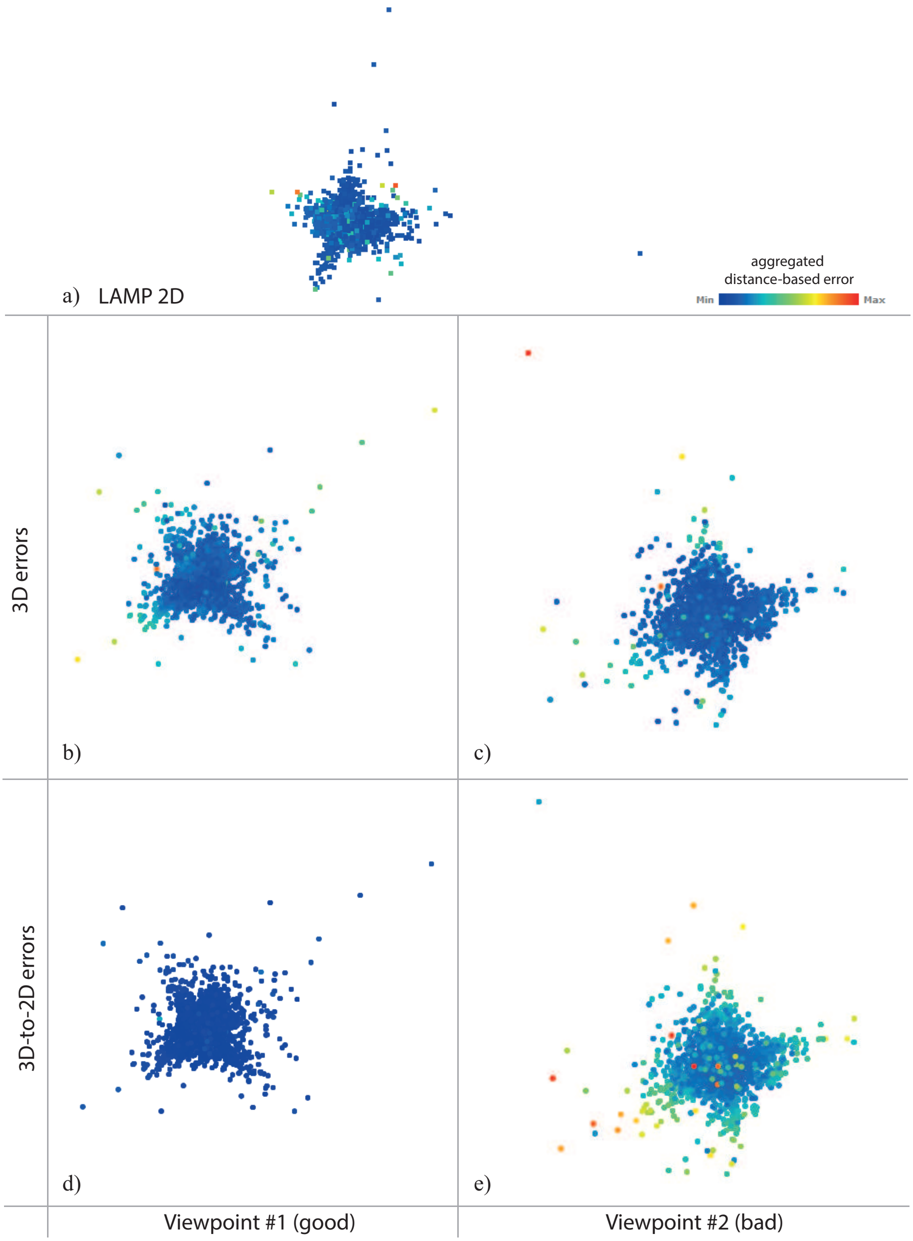


Figure 5.2: Analysis of aggregated distance-based errors of different LAMP projections of the *ALL* dataset: (a) 2D projection errors; (b,c) 3D projection errors for two different viewpoints; (d,e) 3D viewpoint-dependent projection errors, for the same viewpoints shown in images (b,c) respectively. See Section 5.1.1.

We can see then, by looking at Figs. 5.2b–c, that for this specific combination of dataset and projection (our running example – *ALL* and LAMP), the errors of the 3D projection behave in a very similar way to the errors of the 2D projection earlier shown in Fig. 5.2a: We see a very good distance preservation (low error) in the center of the projection; some high-error outliers in the periphery; and mid-level errors towards the tips of the point clusters visible in the projection.

**3D viewpoint-dependent projection error:** Given the above observation, what is, then, the added-value of using a 3D projection in this case, as compared to a 2D projection? To answer this question, let us consider the key factor that a 3D projection is not a *static* image, but a potentially infinite set of 2D images, where each image is generated by drawing the computed (fixed) 3D projection from a user-specified *viewpoint*. As such, the actual projection error, in the sense of the discrepancy *perceived* by an end user between the (invisible) inter-point distances in the original high-dimensional space and the visible inter-point distances shown on the computer screen, is a function of both the 3D projection *and* the viewpoint from which this projection is further viewed. To capture this 'composition' of the $n$D-to-3D and 3D-to-2D projections involved in generating the final image, we introduce a different way of calculating $e_i^m$ for a specific viewpoint of a 3D projection: Instead of using the 3D inter-point distances to compute the projection error, we first project, for a given viewpoint, the 3D points in the 2D screen plane, and next use the points' 2D positions to compute the projection errors. In other words, our projection error applies Eqn. 5.1 with $m = 2$, and with the 2D point positions given by the concatenation of the $n$D-to-3D projection (LAMP) and the 3D-to-2D viewpoint-dependent projection. The effect of this new type of error computation is that the error of each point changes interactively with the chosen viewpoint. As such, the errors we see, at any moment, are the actual errors of the picture that is being shown on the screen as compared to the high-dimensional data.

Figures 5.2d–e show the errors computed with this new method (3D viewpoint-dependent errors) for the same viewpoints as shown in Figs. 5.2b–c. The obtained error plots shown in Figs. 5.2d–e are quite different from the 3D errors shown in Figs. 5.2b–c and also from the 2D projection errors shown in Fig. 5.2a. For instance, the error shown for the viewpoint corresponding to Fig. 5.2d is considerably lower than all other errors shown in Fig. 5.2. In other words, this specific view of a 3D projection of the analysed dataset reflects the high-dimensional inter-point distances best from all considered projections. For instance, the viewpoint shown in Fig. 5.2e shows that the displayed errors are worse than any other shown in the other considered projections and/or viewpoints thereof – we see high-error outliers mixed in the central area, and mid-to-high-error points all around the periphery.

From the insights presented in Fig. 5.2, several conclusions can be drawn. First and foremost, we see that 3D projections have a *measurable* added-value as opposed to the classical 2D projections studied in Chapter 3: If viewed from a suitable viewpoint, such 3D projections show lower errors than corresponding 2D projections using the same projection technique and executed for the same dataset. In other words, it can be better (error-wise) to use an $n$D-to-3D projection, followed by the choice of a suitable 2D viewpoint, than to use a direct $n$D-to-2D projection. However, a direct consequence of this is that the choice of viewpoint can also *negatively* influence the result, in terms of both its error and the insight it can convey: Depending on the choice of such viewpoints, one can achieve lower, but also higher, errors as compared to a 'static' viewpoint-independent 2D projection. Thirdly, we see that 3D projections clearly suffer from a given amount of occlusion – that is, not all observations will be equally well visible from any chosen viewpoint. Therefore, depending on the analysis task at hand, certain viewpoints of the same 3D projection may be better (or worse) than other viewpoints. The main conclusion of this error analysis of 3D projections is, thus, that 3D projections do have the potential of showing better insights into the original high-dimensional dataset than static 2D projections, *but* such insights are dependent on the choice of a suitable viewpoint. As such, we will next explore the design of explanatory mechanisms that show how users can choose good viewpoints of 3D projections for exploring specific aspects of the data at hand.

We next show how such raw 3D scatterplots can be enhanced with visual explanatory tools to address the above-mentioned questions regarding the connection of the visible structures in the scatterplot with the original high-dimensional attributes. These tools include enhanced biplot axes (Sec. 5.1.3), enhanced axis legends (Sec. 5.1.4), and a viewpoint legend (Sec. 5.1.6). Following a brief overview of the explanatory goals we aim to address (Sec. 5.1.2), these tools are explained next.

### 5.1.2 Attribute exploration in 3D projections

Refining the explanatory goals outlined at the beginning of this chapter, we identify the following aspects which we aim to address for a 3D projection (for details, see our original publication [35]):

1. explain the dimensions of the 3D projection space based on the original $n$ high-dimensional attributes;
2. explain the distances between projected 3D points based on the corresponding distances between the same points, in the original $n$D space;
3. support users in choosing viewpoints that are good for addressing given data-exploration tasks;
4. compare how good 2D and 3D projections are, relative to each other, in supporting specific data-exploration tasks;

As introduced in Sec. 2.5, these aspects are currently addressed by a number of visualization techniques. We briefly overview these techniques and their limitations in the context of, and with a focus on, explaining 3D multidimensional projections.

*Aspect 1* is a well-known, and much, studied, problem in visualizing multidimensional data by means of dimensionality-reduction methods. Arguably the earliest methods to address this aspect are *biplots* [71, 69]. Biplots can be best understood by comparing them with classical scatterplots: In a classical scatterplot, data (having two or three dimensions) is projected along two, or three, perpendicular axes, respectively. This way, one can directly read the values of the original variables along the respective axes, much like in a Cartesian plot. In contrast, biplots achieve the same goal (linking the positions of the projected observations with values of their variables) by adding, to the plot, depictions of the values of the variables of the projected dataset. As Gower mentions [69], this allows correlating observations with each other, and also correlating observations with values of their variables. The distribution of the projected variable values are called *biplot axes*, by analogy to the (Cartesian) axes of a classical scatterplot. In this model, there are, thus, as many biplot axes as original high-dimensional variables. However, in contrast to Cartesian scatterplots, these biplot axes need not be, and they usually are not, orthogonal to each other – indeed, the original variables need not be (fully) independent. When shown in a visualization, biplot axes help seeing the way the original high-dimensional variables change in the projection space, in the same way that Cartesian scatterplot axes do.

Biplot axes have been proposed for linear projections, in which case these axes become straight lines in the projection space [71, 1]. While useful, this case does not cover the more general nonlinear projections, which, as we have outlined earlier, have in general lower projection errors than their linear counterparts, *e.g.* [95, 141]. Additionally, the construction of biplot axes currently known in the literature assumes that one knows the internals of the projection technique, *e.g.* singular value decomposition [71]. Constructing biplot axes for *any* type of projection is, thus, not handled by this approach.

Besides biplots, aspect 1 can be approached by the techniques proposed by Broeksema *et al.* [24] and Oeltze *et al.* [132]. Instead of showing the directions of maximal variation of the $m$ high-dimensional variables in the projection space, this alternative approach shows the amount of variation of these $m$ variables along the $x$ and $y$ axes of the 2D projection space. These amounts, also called *loadings* [71, 1], are essentially equal to the projections of the $m$ (straight-line) biplot axes on the two largest eigenvectors of the observation covariance matrix that determine the $x$ and $y$ axes of a 2D projection [24]. Loadings are visualized by two bar charts, or legends, having, each, $m$ bars, each bar indicating the loading of a variable on the corresponding screen axis. A third bar chart indicates the amount of variance of the $n$ variables which is not captured by the two eigenvectors that determine the 2D projection. By identifying the largest bars in the legends for the $x$ and $y$ axes, one can therefore determine which variables mainly explain the spread of the points in the 2D projection. However, this technique has not been extended for 3D projections, whose $x$ and $y$ screen axes are determined by the chosen viewpoint. Separately, and similarly to biplot axes, the aforementioned axis legends have been so far only used in the context of linear projections.

*Aspect 2*, *i.e.*, explaining the distances between projected points in terms of the corresponding distances between high-dimensional points, is covered by various techniques that include error metrics (such as stress (Eqn. 2.2), correlation [63], neighborhood-preservation plots [139]); and, separately, user studies aiming to show how users can reason about the original high-dimensional distances when seeing only the low-dimensional distances in the projection space [113]. Global aggregate error metrics are good in comparing different projections on a high level, *e.g.*, to determine whether a projection technique generates more accurate results than another projection technique. However, such global metrics cannot show how errors are distributed in a projection, *i.e.*, which parts thereof are the most severely affected by distance distortions. As such, they are less useful for exploratory visualization. A more fine-grained explanation of distance preservation is offered by distance scatterplots, which plot all distances between point-pairs in $n$D *vs* the corresponding distances in $m$D [95]. The deviation of such plots from a straight diagonal line (the ideal case) show how distances are increased, respectively decreased, by the projection technique. However, distance scatterplots do not link distance errors with observations, *i.e.*, cannot show for which subsets of points in a projection large (or small) errors occur. On an even finer level of detail, errors can be explainer by local metrics which show how (small) neighborhoods around projected points are affected by the projection technique. Such local explanation methods include the projection precision score (*pps*) [159], distance stretching and compression metrics [8], the distance error metrics introduced by us in Chapter 3), and the neighborhood-preservation metrics (Chapter 4). Such metrics are discussed in more detail in [14]. However, to our knowledge, all these error metrics have been so far only constructed and applied for 2D projections. In Section 5.1.1, we showed how such metrics can be generalized for 3D projections. Apart from this, it is important to note that projection errors are useful to show *where* in a projection problems appear; however, they do not explain *why* points are projected the way they are. As such, projection error metrics are indeed useful to show whether a projection is accurate or not – but, for an accurate projection, they cannot further explain the meaning of the resulting point-distribution patterns.

*Aspect 3*, *i.e.* choosing one or several viewpoints (in the case we have a 3D projection or scatterplot) so that specific questions are answered by these, can be addressed in different ways, as follows. First, 3D scatterplots can be further explained by adding three supplementary 2D views (following a linked-view metaphor). Selecting and/or brushing points between the linked views allows explaining complex patterns in the 3D projection by using the relatively simpler 2D views [146]. The same aspect is also addressed by offering users enhanced mechanisms to manipulate the projection in terms of changing the viewpoint [50, 158], smoothly navigating between different viewpoints [87, 86], and allowing the user to interactively arrange the variable axes in the projection [34]. However,while simplifying the exploration, such mechanisms are not tuned to tell the user which aspects related to the original high-dimensional variables one can see in a given viewpoint, nor do they give a compact overview of all such aspects that all possible viewpoints of a 3D projection can provide.

*Aspect 4*, *i.e.* choosing between the use of 2D *vs* 3D projections for a given application or end-to-end task, is arguably one of the hardest problems involving the estimation of usability of 3D projections. Given the very high level of this question, it is not surprising that the instruments employed to address it center, most often, on user studies and evaluations (see further Sec. 2.5). While the insights provided by such studies are of clear added value in terms of understanding the general pro's and con's of 2D *vs* 3D projections, they are generally limited to high level, qualitative, explanations. Quantitative and detailed insights, showing *e.g.* why, where, and how much do errors in a 2D projection differ from those in a 3D projection, are lacking. Providing such detailed insights can arguably help both users and designers of projection-based visualization tools in selecting the best technique for a specific exploration task.

From the above, we conclude that there is still significant open space for improvement in the context of explaining 3D projections of multidimensional data, in order to make these more useful and usable. Separately, as outlined in Sec. 5.1.1, and in line with earlier evidence discussed above, 3D projections do have the potential of showing data with fewer errors, if suitable viewpoints are chosen. In the remainder of this chapter, we present several mechanisms that aid the above-mentioned explanatory goal.

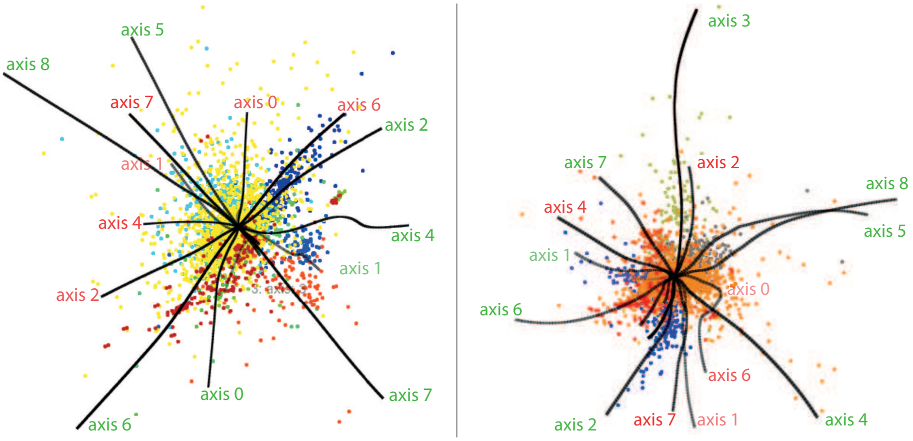
### 5.1.3 Generalizing biplot axes

As explained in Sec. 2.5, and also outlined above, biplot axes are typically constructed by applying singular value decomposition (SVD) to the high-dimensional observations (Eqn. 2.3). The eigenvectors of this decomposition represent the biplot axes, which are next drawn (using *e.g.* vector glyphs) atop of the projected points [1]. While this technique is extremely simple, it has several limitations. Conceptually speaking, constructing (and drawing) biplots by applying SVD to all the points only makes sense if we use the same technique (SVD) to also *project* the points. In other words, this way of constructing biplot axes is only applicable if the projection technique used is a linear one. This, obviously, precludes all non-linear projection techniques from using such biplot axes – indeed, if we were to *e.g.* superimpose SVD biplot axes atop of a LAMP projection [95], or any other non-linear technique discussed in Sec. 2.3.2, the result would make no sense, since the axes, respectively the projected observations, would not match each other. A separate problem is that the SVD-based construction of biplot axes assumes that a single global transformation is used to project all observations. If this is not the case, *e.g.*, if different transformations are used to project subsets of points, such as in PLMP [141], then we cannot use a global computation of biplot axes, for the same reasons as those explained above for non-linear projections.

If we were to generalize biplot axes from the classical SVD ones to any projection technique, a second problem appears. For SVD, such axes are simple to compute since we, obviously, have access to the exact details of how the projection is done (specifically, the matrices $\mathbf{U}$, $\Delta$, and $\mathbf{V}$ discussed in Sec. 2.3). If we were to generalize this to other techniques, we would need, by analogy, to 'open up' the implementation of the respective techniques, and extract the required information needed to compute the biplot axes. While this can be done, it would imply that one would need to have a separate implementation of biplot axis construction for each projection technique out there. Clearly, this is not a desirable situation, given the large (and increasing) number of projection techniques that exist, and are added to, the literature.

Given the above, we propose to generalize the construction of biplot axes by taking a different approach than the one proposed for linear projections. Specifically, we consider the ability of the projection function $f$ (see Sec. 2.3 to project any $n$D point to $m$D. Given such a function, chosen to project the regular observations, we reuse it to create the biplot axes. For this, we create a number of $S$ artificial observations $\mathbf{p}^i_{1 \leqslant j \leqslant S}$, for each high-dimensional variable $i$ of our dataset. These observations are created as follows: The values for dimension $i$ of the points $\mathbf{p}^i_j$ are uniformly distributed over the range of variable $i$, with equal steps; the values of all other dimensions $k \neq i$ are set to the average of the values that dimension $k$ takes in our dataset. In other words, we augment the set of observations that is taken as input by $f$ by a number of $S \cdot n$ points, which are basically uniformly sampling the $n$ high-dimensional axes, clipped by the $n$-dimensional bounding-box determined by the variables' ranges, and constructed so as to pass through the centroid of this bounding-box. Next, we project all these additional points $\mathbf{p}_j^i$ to 3D, yielding a set of three-dimensional projections $\mathbf{q}_j^i$. Finally, for all points $\mathbf{q}_j^i$ belonging to the same axis $i$, we draw a polyline $c_i$ that connects them. This polyline represents the generalized biplot axis for dimension $i$. In practice, we use a value of $S = 100$ sample points.

The resulting generalized biplot axes are shown in Fig. 5.3 a for the same *ALL* dataset, projected by LAMP in 3D, which was discussed earlier in this chapter. Several aspects are visible here. First, we see how the generalized biplot axes intersect in (or close to) the projection's centroid, and span the extent of the projected points, as expected. In this sense, the axes resemble typical Cartesian axes used in, *e.g.*, classical scatterplots. Following this analogy, we can use the lengths of the generalized axes to reason about the spread of data values along specific dimensions – short axes indicate variables which have a smaller range, and long axes indicate variables having a larger range. The directions of variation of variables along their respective axes is indicated by colored labels placed at the axes' ends, with green indicating the maximum, and red indicating the minimum, of a variable, respectively.



a) LAMP Projection

b) FBDR Projection

Figure 5.3: Generalized biplot axes for the *ALL* dataset projected by LAMP (see Fig. 5.1).

The curvature of the generalized axes indicates the local spatial deformations that the projection technique achieves – simply put, deviations from a straight line indicate that the projection is non-linear in the respective areas. More interestingly, the angles formed by axes indicate correlations between variables: Axes which form close angles indicate strongly correlated variables. Direct or inverse correlation is easily read by looking at how the orientations of the respective axes matches (or not). Similarly, axes that form large angles (with a maximum of 90 degrees) indicate highly uncorrelated, or independent, variables.

As an additional example, Figure 5.3 b shows a different projection of the *ALL* dataset. Here, in contrast to LAMP, we use the Force-Based Dimensionality Reduction (FBDR) technique [179]. We selected FBDR as a projection technique on purpose, as we were aware of its high nonlinear nature, and wanted to confirm this insight. Visualizing the generalized biplot axes in Fig. 5.3 b confirms our impressions. Indeed, the biplot axes look now significantly more curved than in the LAMP projection. This is an useful insight that would tell a user that using FBDR to reason about the correlation of variables is not the optimal choice, LAMP being much better for this task. However, this does not imply that FBDR cannot generate useful projections. For instance, it could be so that FBDR minimizes the various projection errors discussed in Chapter 3 better than LAMP, in which case the FBDR projection would be better than the LAMP one for tasks that center on finding groups of similar points. Checking this can be easily done by visualizing the projection errors, as explained in Sec. 5.1.1 (omitted here for brevity).

### 5.1.4 Explanatory axis legends

To explore 3D projections, we provide standard viewing-control tools, in our case a so-called virtual trackball widget (controlled by the mouse) to rotate the viewpoint, zoom in and out of the data, and translate the viewpoint in the view plane (pan the view).

One key difficulty that arises for both 2D and 3D projections of multidimensional data, is to convey a (simple) understanding of the meaning of the screen axes. Indeed, typical end users are accustomed with Cartesian plots, where these axes have a clear meaning: In the case of 2D plots, these axes map two data variables; in the case of 3D plots, the screen axes do not directly map three data variables (given that the user can choose any viewpoint). However, axis legends embedded in the visualization give this insight. In our case of viewing 3D projections of high-dimensional data, we have both difficulties.

To address this, we extend an earlier technique proposed to explain the meaning of the screen axes for 2D projections [24]. In this technique, the two screen axes $x$ and $y$, which we further denote by $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively, are annotated by two barchart legends. Each legend has as many bars as original high dimensions. The length of a bar $j$ in the legend for $\mathbf{x}_i$ indicates, intuitively put, how well can one see the variation of variable $j$ along screen axis $\mathbf{x}_i$. A third legend is added to show which variables can be worst seen along both screen axes $\mathbf{x}_1$ and $\mathbf{x}_2$, as they are not captured well by the used 2D projection. For full details, we refer to [24].

By analogy, our 3D projection case has the same problems as the 2D projection case in [24]. Specifically, given a viewpoint defined by screen axes $\mathbf{x}_1$ and $\mathbf{x}_2$, and also by a viewing direction $\mathbf{x}_3$, our users' questions are essentially identical to those treated in [24]: What is the meaning of the screen axes $\mathbf{x}_1$ and $\mathbf{x}_2$ in terms of the original variables? Which original variables are hard to explore from the given viewpoint? Summarizing the above, we aim to provide additional visual details that explain what a given viewpoint can show.

Following our analogy, we approach the explanatory problem by using three barcharts, or axis legends (Fig. 5.4). Each legend corresponds to one of the axes $\mathbf{x}_i$, $1 \leqslant i \leqslant 3$, and has as many bars as high-dimensional variables in our dataset. The length of the $i^{th}$ bar in the barchart for axis $j$ indicates how well the axis $\mathbf{x}_j$ aligns with the variation of variable $i$. Conceptually speaking, this is the same metric as the one used by [24] in their 2D barchart legends. However, our actual metric used to scale bars is different, for a number of reasons. First and foremost, we use the underlying projection technique as a black box, and as such, do not have access to quantities such as loadings, which were used in [24] to compute the bar lengths. Moreover, it is not clear if such quantities can be computed in general for any (non-linear) projection technique. Secondly, we have a 3D projection, and as such, we can (and should) treat the construction of legends for all three axes $\mathbf{x}_j$ similarly. This was not the case in [24], where a 2D projection was used.

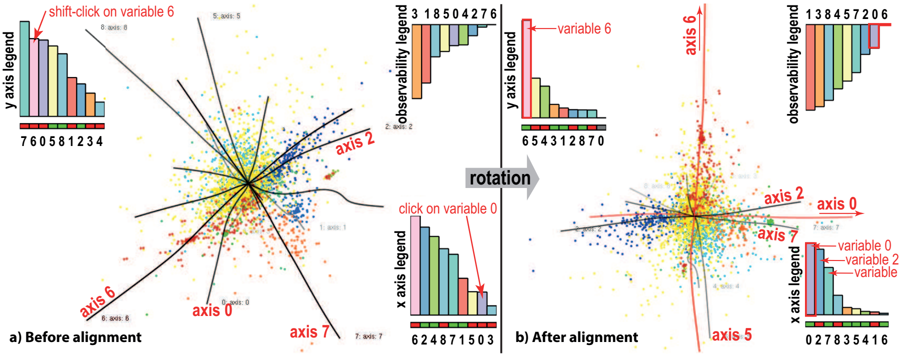


Figure 5.4: Axis barcharts for the *ALL* dataset projected using LAMP. Left view: arbitrary viewpoint, obtained by free rotation. Right view: aligned configuration obtained from the left one by aligning variable 0 with the $x$ axis, followed by aligning variable 6 with the $y$ axis.

Given the above, we chose to compute the above-mentioned legend-bar lengths by following a simple intuition, *i.e.*, by measuring how well a generalized biplot axis $c_i$ aligns well with a screen axis $\mathbf{x}_j$. This measure yields the length of bar $j$ in the barchart of axis $\mathbf{x}_j$. Formally, the above measure is given by the absolute value of

$$h_i^j = \left( (\mathbf{q}_S^i - \mathbf{q}_1^i) \cdot \mathbf{x}_j \right) \left( 1 - \frac{|\|c_i\| - \|\mathbf{q}_S^i - \mathbf{q}_1^i\||}{\|c_i\|} \right) \tag{5.2}$$

where $c_i$ is the biplot axis for variable $i$ (see Sec. 5.1.3) and $\|c_i\| = \sum_{j=1}^{S-1} \|\mathbf{q}_j^i - \mathbf{q}_{j+1}^i\|$ denotes the length of the curve $c_i$. The explanation of Eqn. 5.2 is as follows: The term $(\mathbf{q}_S^i - \mathbf{q}_1^i) \cdot \mathbf{x}_j$ gives the length of $c_i$ along $\mathbf{x}_j$. If this value is high, it means that we can easily *see* how variable $i$ spreads (varies) along $\mathbf{x}_j$, and we thus indicate this by a long bar $i$ in the legend of $\mathbf{x}_j$. Note that, if this value is low, it implies that either biplot axis $i$ is far from being aligned with screen axis $\mathbf{x}_j$ (in which case, we can try to obtain a better view by choosing a different viewpoint); or that the range of variable $i$ is very small, as compared to the other variables in the dataset (in which case, we cannot obtain a better 'view' of this variable; and it can also be argued that this is not useful, since the variable's range is very small). The term $1 - \frac{|\|c_i\| - \|\mathbf{q}_S^i - \mathbf{q}_1^i\||}{\|c_i\|}$ in Eqn. 5.2 is a measure of how straight the generalized biplot axis $c_i$ is, obtained by comparing the actual length $\|c_i\|$ of the curved axis with that of the straight-line segment $\mathbf{q}_S^i - \mathbf{q}_1^i$ that connects its endpoints. The intuition behind adding this term to the computation of $h_j^j$ is that reading a variable that is projected along a *straight* biplot axis, by using a straight screen axis $\mathbf{x}_j$ is, arguably, much easier than using a straight screen axis to read a variable projected along a curved biplot axis.

The interpretation of bar lengths in our three legends is identical to those in [24]: Long bars in the legends of the $x$ and $y$ screen axes are good, as the indicate we can read the respective variables easily along these axes. In contrast, long bars in the legend for the view-direction axis $\mathbf{x}_3$ are to be avoided, as they show that important data variations in our dataset are not visible, or not observable, from the current viewing direction. As such, the barchart for $\mathbf{x}_3$ is next called the *observability legend*. In contrast to the original barchart design in [24], where all bars were growing upwards in their respective charts, we choose to let bars in the observability legend grow downwards (see Fig. 5.4). This makes reading the three legends more uniform: The upward direction represents a desirable situation (good observability); the downward direction represents an undesired situation (bad observability).

The axis barcharts presented above, *i.e.*, the values of $|h_i^j|$, show how well a variable $i$ aligns to a screen axis $\mathbf{x}_j$. However, they do not show how to read the values of variable $i$ along $\mathbf{x}_j$, *i.e.*, if variable $i$ increases in the same sense, or in the opposite sense, of $\mathbf{x}_j$. This insight can be obtained by looking at the colored text labels attached to the endpoints of the biplot axis $c_i$. However, often, such labels can be easily occluded in a 3D plot. Hence, we add a supplementary cue, in terms of a small icon placed under each bar. The icon is colored green if variable $i$ increases along the vector $\mathbf{x}_j$ (*i.e.*, if $h_i^j > 0$); red if variable $i$ decreases along $\mathbf{x}_j$ (*i.e.*, if $h_i^j < 0$), and gray if variable $i$ varies in a direction orthogonal to that axis (*i.e.*, if $h_i^j = 0$). Icons are not used for the observability legend, since this legend shows data variations which are not observable from the current viewing direction. A variation of the above technique is to color the icons on the actual value of $|h_i^j|$, as follows: Icons where $h_i^j > 0$ are colored by interpolating between gray ($h_i^j = 0$) and saturated green ($h_i^j = 1$); and icons where $h_i^j < 0$ are colored by interpolating between gray ($h_i^j = 0$) and saturated red ($h_i^j = -1$). While this gives a more continuous, and nuanced, view of how easy is to read a variable's variation in the positive, respectively negative, sense of a screen axis, it also creates, we believe, more interpretation difficulties. As such, we chose for the simple three-color categorical mapping described above.

Finally, to make the interpretation of the barcharts easier, we label them by the variable names, and color them by using a categorical colormap generated using the well-known ColorBrewer tool [74].

A subsequent degree of freedom relates to how bars are sorted in a legend. We explored two designs here. In the first designs, all bars are sorted based on the same criterion in all legends, *e.g.*, alphabetically on their variable names. This places the bar for a given variable at the same positions in all three legends. This is desirable when, in the current exploration scenario, we already have a few important variables which we know, *a priori*, we want to explore. As bars have fixed positions in the three legends, we can then easily visually correlate their lengths. This is the same design as originally proposed by [24]. However, a more interesting (and we believe, useful) scenario is the situation when all variables are, *a priori*, of equal interest. In that case, it is natural to (want to) focus on the variables which are best visible from a given viewing direction. To emphasize these, we sort all three legends in decreasing order of their respective bar lengths. This is the mode shown in Fig. 5.4 and all subsequent figures in this chapter. Finding which variables are best observable from the current viewpoint is now easy – one simply looks at the longest (*i.e.*, leftmost) bars in the legends for the $x$ and $y$ screen axes. Conversely, seeing which variables one should not attempt to analyze from the current viewpoint is equally easy – one looks at the longest (*i.e.*, leftmost) bars in the observability legend. The sorted mode has the extra added-value that it lets our visualization scale to handle datasets with tens of dimensions or more: To make bars and their annotations visible, we limit ourselves to displaying only the $n_{max} = 20$ longest bars in any legend. Arguably, this is the best one can do given this visual design, in the sense that we explain to the user the most salient (observable) $n_{max}$ dimensions. As a subsequent design element for the barcharts, we adopt the linking mechanisms introduced in [24]: Brushing a bar highlights the corresponding dimension (if present in the $n_{max}$ displayed ones) in all three legends. As a final design element, we use the observability values ($|h_i^3|$) to control the transparency of the biplot axes. This way, axes which are poorly observable from the current viewpoint (large $|h_i^3|$ values) become more transparent, and thus also clutter the visualization less (see *e.g.* 7,6,2 in Fig. 5.3a). Conversely, axes which are well observable from the current viewpoint (small $|h_i^3|$ values) become more opaque, and thus attract the attention of the user (see *e.g.* axis 5 in Fig. 5.3b).

Arguably the largest design difference between our barcharts and those proposed in [24] relates to i*nteractivity*. In our case, changing the viewpoint by manipulating the virtual trackball changes the values of the three axes $\mathbf{x}_j$. This invokes a recomputation and redisplay of all barcharts (using Eqn. 5.2). This way, one can dynamically monitor how specific variables map to the screen axes while one changes the viewpoint, and thus, decide whether the current viewpoint is a good or interesting one, or if one, for instance, wants to go to a previous viewpoint. This feature showed up to be especially useful when fine-tuning a viewpoint to show specific variables of interest.

### 5.1.5 Aligning axes

As noted several times so far, an important task in the exploration of 3D projections is to study how specific variables correlate (or not) with each other. Doing this by interactively manipulating the viewpoint legend, as outlined at the end of the previous section, is possible, but can be time-consuming. Indeed, it can be hard to manually select the precise viewpoints that best show the variation of a specific subset of axes.

To help this, we provide an automatic alignment mode, as follows. To align a variable $i$ with any of the $\mathbf{x}_j$, $j \in \{1, 2\}$ screen axes, we click the respective bar for that variable in the respective barchart. This initiates a smooth rotation that sets the viewing direction to one where $h_i^j$ is maximal. This leaves us an additional degree of freedom, namely, the rotation of the viewpoint around $\mathbf{x}_j$. We exploit this to allow the alignment of a second variable with the remaining screen axis, by shift-clicking another bar in the barchart of this screen axis. Two such clicks, thus, allow one to smoothly change the viewpoint from its current setting to the one where two user-chosen variables are best aligned with the two screen axes $\mathbf{x}_1$ and $\mathbf{x}_2$. This way, we can interactively create scatterplots of arbitrary pairs of such variables with only two clicks. It should be noted, however, that such scatterplots are *not* identical to the traditional (Cartesian) ones that one would create by simply considering two variables from a multidimensional dataset. Indeed, our two-variable scatterplots are limited, in terms of accuracy, by the underlying *fixed* 3D projection. While this accuracy is, in general, lower than when considering two-variable Cartesian scatterplots, the added value of constructing them from projections is that we can *smoothly* navigate between *any* pair of such scatterplots, simply by performing a viewpoint rotation. Doing this navigation for Cartesian scatterplots is possible, but involves more complex deformations, which make it harder for users to maintain their mental map during the navigation [50].

Figure 5.4 shows our mechanism for scatterplot construction by iterative axis alignment. Starting from an arbitrary viewpoint (Fig. 5.4a), we next click on the bar of variable 0 in the $x$ axis legend and next shift-click on the bar of variable 6 in the $y$ axis legend to obtain the aligned view in Fig. 5.4b. Looking now at the longest bars in the two legends, we see that the spread of points along the $y$ screen axis is mainly due to variable 6; in contrast, the spread along the $x$ screen axis is due to the combined effect of variables 0, 2, and 7. Since the lengths of the bars for these three variables are roughly similar in the $x$ legend, it follows that the respective three variables are also strongly correlated. Looking at the colors of the icons below these bars in the $x$ legend, we discover that variables 0 and 7 are directly correlated, and both are inversely correlated with variable 2.

### 5.1.6 Viewpoint legend

As outlined above, the interactive manipulation of the viewpoint allows us to freely explore the space of possible views of our 3D projection. Interesting viewpoints being discovered are highlighted by specific distributions of the bar lengths in the three axis barcharts. At the other end of the exploration spectrum, going to a targeted viewpoint which allows optimal exploration of two given variables, is provided by the mechanism outlined in Sec. 5.1.5.

However flexible, the above two exploration mechanisms do not cover the full spectrum of possible use-cases. Indeed: The axis-alignment tool is good for targeted navigations, when we know the variable-pair we are interested in, but not for additional explorations. In contrast, free-mode rotation is good to discover new insights, but navigating through *all* such possible viewpoints is clearly prohibitive.

We next describe a mechanism, called the *viewpoint legend*, that aims to bridge the preciseness (but narrow focus) of the axis-alignment tool with the freedom (but vagueness) of free navigation. Specifically, we aim to provide a compact summary of all variable-pair relations that one could discover, using free navigation, if one had the time to explore all possible viewpoints. The viewpoint legend consists of two elements. The first one is a sphere $S$ of center $\mathbf{c}$ (Fig. 5.5a) whose points $\mathbf{v} \in S$ describe all view directions $\mathbf{c} - \mathbf{v}$ that one can look at a given 3D projection, modulo zooming and panning. The current viewpoint is indicated on the sphere by a cross and is, by construction, always located in the center of the sphere's image. For each viewpoint $\mathbf{v} \in S$ we define its *ability* of showing the variable-pair $(i, j \neq i)$ as

$$q(\mathbf{v}, i, j) = \|(h_i^1, h_i^2) \times (h_j^1, h_j^2)^{\mathrm{T}}\|. \tag{5.3}$$

More formally, $q$ measures how well a 3D projection, viewed from $\mathbf{v}$, shows the variation of variable $i$ *vs* $j$, modulo rigid transformations such as scaling (zooming), translation (panning), and rotation of the view around the viewing direction $\mathbf{c} - \mathbf{v}$. Large values of $q(\mathbf{v}, i, j)$ indicate that the biplot axes for variables $i$ and $j$ have long projections onto the view plane *and* that these projections form a large angle (maximally, $90^\circ$). In other words, large values of $q$ tell the existence of viewpoints from which we can construct good two-dimensional-like scatterplots showing pairs of (largely) independent variables which also have significant variations.

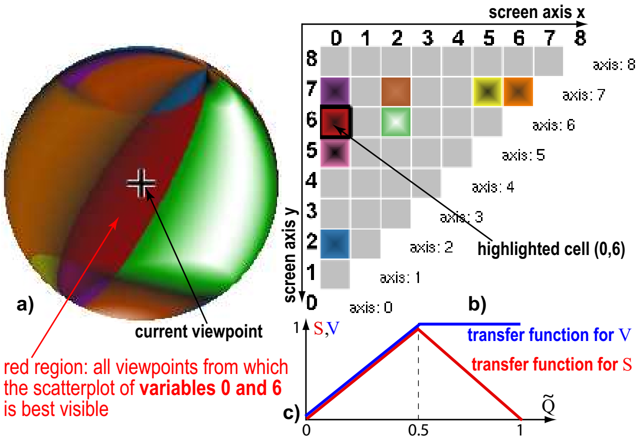


Figure 5.5: Viewpoint legend for the configuration shown in Fig. 5.4 b.

As indicated by Eqn. 5.3, several variable-pairs may be well visible from a given viewpoint $\mathbf{v}$. Showing all this information to the user may be possible, but overwhelming. Instead, we choose to show the *best* visible such variable pair. To find this, we compute

$$Q(\mathbf{v}) = \max_{1 \leqslant i \leqslant n, 1 \leqslant j \neq i \leqslant n} q(\mathbf{v}, i, j), \tag{5.4}$$

and, further, normalize this value over all viewpoints to yield

$$\bar{Q}(\mathbf{v}) \in [0, 1] = Q(\mathbf{v}) / \max_{\mathbf{u} \in S} Q(\mathbf{u}). \tag{5.5}$$

When computing $Q(\mathbf{v})$ (Eqn. 5.4), we also compute the pair of variables $p(\mathbf{v}) = (i, j)$ that yields this maximum value, *i.e.* $p(\mathbf{v}) = \arg\max_{1 \leqslant i \leqslant n, 1 \leqslant j \neq i \leqslant n} q(\mathbf{v}, i, j)$. Having this information, we compute the set $P = \{p(\mathbf{v})\}$ containing the $C$ variable pairs that have maximal values $\bar{Q}(\mathbf{v})$ for all viewpoints $\mathbf{v} \in S$. In other words, $P$ tells us which are the

$C$ best-visible variable-pairs from all possible viewpoints. In practice, we limit $C$ to a maximal value of 8, which yields thus maximally 8 such variable-pairs. Hence, we can visualize these pairs over $S$ by categorical color mapping, *i.e.* by assigning to each $p \in P$ a distinct color.

The above procedure does not, however, cover all points $\mathbf{v} \in S$, but only those from which one of the $C$ best visible variable-pairs shows up. For all other points of $S$, we color them gray, to indicate that none of the overall-best-visible variable-pairs is visible from there. As a final design, we set the saturation $S$ and luminance $V$ of each sphere's colored point by $\bar{Q}(\mathbf{v})$, via the transfer functions depicted in Fig. 5.5c. By this, we effectively encode two information elements at each (view)point – the identity of the best-visible variable-pair from there (hue), and how well this variable-pair is visible from there, as compared to how well the same or other variable-pairs are visible from other points (luminance and saturation). The global result, shown in Fig. 5.5 b, is to create a pseudo-shading effect, where viewpoints for highly-visible variable-pairs appear as highlights; and points which cannot show a globally-best-visible variable-pair appear dark, respectively. Note that, although highlights do not carry a hue, finding out which variable-pair they can best show is easy – we simply look at the darker, saturated, colored points surrounding such a highlight. We implement Eqns. 5.3-5.5 by sampling $S$ with about $50 \times 50$ points, uniformly distributed in polar coordinates. The resulting polygonal representation is rendered with standard bilinear color interpolation.

To make the above best-visible variable-pair scheme complete, we need to add a (categorical) color legend. We achieve this by the matrix shown in Fig. 5.5b. Matrix cells map all variable-pairs $p = (i, j)$, whose hues are set exactly as for the sphere points, *i.e.*, if $p \in P$, we use $c(p)$, else we use gray. The brightness of a cell $p$ is set to linearly vary between the default brightness of its color $c(p)$ (at the cell border) and $\max_{\mathbf{v} \in S} q(\mathbf{v}, i, j)$ (at the cell center). This way, the matrix shows two information elements at each cell or variable-pair – first, the color that that variable-pair is mapped to (cell hue, visible at the border); and secondly, the relative visibility of this variable-pair over all viewpoints, as compared to other variable-pairs (cell luminance, visible at the center).

Figure 5.6 shows the working of the viewpoint legend for our *ALL* dataset. Turning the virtual trackball in the main (scatterplot) view rotates the viewpoint sphere, and conversely – turning the viewpoint legend changes the current viewpoint. As such, the viewpoint legend acts both as an output device (showing what can be seen from basically 50% of all possible viewpoints at once) and input device (allowing users to change the viewpoint to go to one where a specific variable-pair is best-visible from). The viewpoint sphere and matrix legend are also linked – the matrix legend highlights the cell for the current viewpoint (so one immediately knows to which variable-pair $(i, j)$ the color under the central cross cursor belongs); and clicking in a matrix cell $(i, j)$ rotates the viewing direction to the viewpoint from which the pair $(i, j)$ is best visible (so one can construct the best-possible two-variable scatterplots $(i, j), \forall i \neq j$ by a single click).

Several use-cases can be described where the viewpoint legend brings in effective exploratory help, as follows:

**What is a good viewpoint to examine $(i, j)$ from?** To find this, one (1) finds the

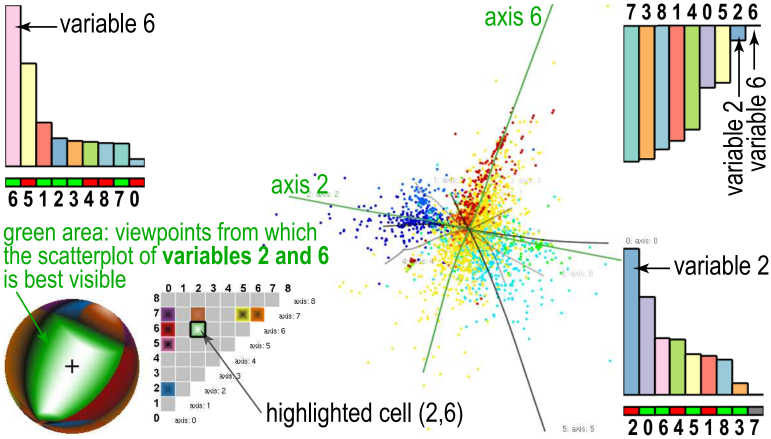


Figure 5.6: Viewpoint widget set to highlight a viewing direction that best shows variables 2 *vs* 6.

cell $(i, j)$ in the matrix legend, memorizes its hue, (2) turns the sphere to see bright highlights surrounded by the memorized hue, and (3) turns the sphere so the cross enters such a highlight. Alternatively, one can directly click on the cell, to go to the best such viewpoint. If the luminance of cell $(i, j)$ is low, however, it means there is no such really good viewpoint available.

**Can I easily study variable $i$ *vs* $j$?** To find this, one (1) finds the cell $(i, j)$ in the matrix legend, memorizes its hue, (2) turns the sphere to see how large (and brightly highlighted) areas colored in that hue are on the sphere. The size and distribution of such areas tells how easy is to create a scatterplot showing variable $i$ *vs* $j$. Intuitively put, if it's easy to rotate the sphere so as to get the cross in such a bright area, then creating the respective scatterplot is easy. Moreover, the larger the respective area highlight is, the less accurate does one need to fine-tune the viewpoint to get good results.

**Is there any good viewpoint for examining $i$ *vs* $j$?** To find this, one can examine the brightness of the cell $(i, j)$ in the matrix legend. A bright cell will tell that there exists at least one suitable examination viewpoint – and clicking the cell goes to it. Separately, examining the entire sphere for the occurrence of areas colored like cell $(i, j)$ tells how easy is to find such a good viewpoint. If the desired cell $(i, j)$ is dark, then no suitable viewpoint for the task exists. This may indicate either a projection problem, or the simple fact that the variables $i$ and $j$ have very small ranges.

**What should I expect to see from a given viewpoint?** To find this, one should (1) look at the highlighted matrix legend cell and (2) infer the row $i$ and column $j$ of that cell to get the best-visible variable-pair from there. Additionally, one should (2) look at the brightness under the cross cursor on the sphere. A large value indicates that, indeed, pair $(i, j)$ outlined above is *well visible* from there; a dark value indicates that, although pair $(i, j)$ is the *best visible* from the given viewpoint, its overall visibility is very poor. As such, one should not expect to see anything interesting in the sense of a variable-pair correlation from that viewpoint.

**How to study the relation of more than two variables?** A single-hue zone, on the viewpoint sphere, shows a 'field of view' from which the same variable-pair $(i, j)$ is best visible; therefore, when the viewpoint crosses the border between two such zones, one goes from best-viewing a pair $(i, j)$ to best-viewing another pair $(k, l)$. Often, as we will see in Sec. 5.2.2, pairs $(i, j)$ and $(k, l)$ share a variable (*e.g.*, $j = k$), which makes such border-zones be good viewpoints from where one can examine the interaction of three variables.

## 5.2 Example applications

To illustrate and text the power of the explanatory techniques presented above in the context of understanding 3D projections in terms of their underlying variables, we studied four different multidimensional datasets. These come from four different domains (agriculture, scientific simulations, image analysis, and software quality assessment. The datasets range between 2300 and 200000 observations, and have between 10 and 19 dimensions. All observation values are quantitative. The covered exploration tasks involve a variety of typical cases frequent in multidimensional (projection) data exploration, such as finding correlated and/or independent variables, explaining point clusters and outliers, assessing projection quality, and comparing the suitability of using a 2D *vs* a 3D projection technique. In terms of the latter, we used in total four projection techniques.

### 5.2.1 The Wine dataset: Finding good projection techniques

This dataset has 6497 12-dimensional points. Each point describes a type of *vinho verde* wine [38] by means of 11 physicochemical properties, such as density, concentration of chloride, concentration of sulfur, pH, and alcohol percentage. The $12^{th}$ variable is a (subjective) human-specified level of quality. The underlying use-case for this dataset is finding how the 11 measured variables correlate with the perceived quality, as a first step into designing automatic classifiers for wine [38].

To use dimensionality reduction, we must fist decide which projection technique is best suited. We consider here three DR methods: FBDR [179], ISOMAP [184], and LAMP [95] to project our dataset to 3D (other DR methods can be equally easily used, if desired). Figure 5.7(left) shows the resulting three projections, aligned to be viewed from (roughly) the same viewpoint. The right column in the same figure shows the projections colored by their 3D aggregated distance errors. One way to choose a good projection to work with further would be to select the method that minimizes the errors (Eqn. 5.1). However, as shown by Fig. 5.7(right), these three state-of-the-art projection techniques yield quite similar distributions of error values, so such aggregate errors are not discriminatory enough in this specific case.

In contrast to several of the datasets discussed in Chapter 3, projection errors do not

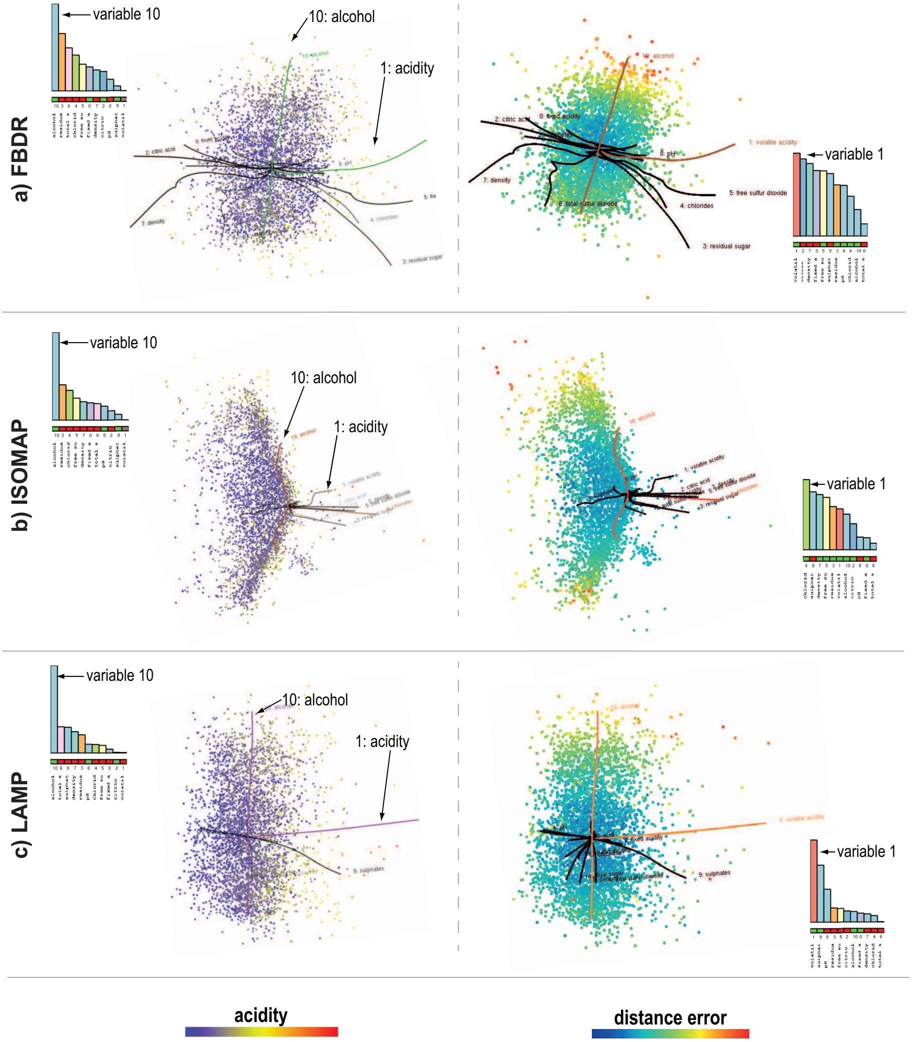


Figure 5.7: Comparing three projection techniques (FBDR, ISOMAP, LAMP) using biplot axes and axis legends (left) and projection errors (right). The selected viewpoint best emphasizes the correlation of the *alcohol* and *acidity* axes. See Sec. 5.2.1.

help us much here in deciding which is the best projection. To further discriminate them, we use the biplot axes (Fig. 5.7(left)). We see here that LAMP creates much straighter axes than FBDR and ISOMAP. Reading data values along such axes is considerably easier than along highly curved axes, as discussed earlier in Sec. 5.1.3. This is an argument in favor of further using LAMP, if we are interested in exploring such our values along axes, much like in classical scatterplots.

However, there are other tasks we would like to accomplish using the selected projection, such as studying correlations of two variables. As a test, we consider the variables *alcohol* and *acidity* (10 and 1, respectively, in Fig 5.7). To best view the correlation of these two variables, we align all projections by clicking on the two bars in the $x$ and $y$ legends, respectively (see Sec. 5.1.5). Additionally, we color all points by their *acidity* values, using a three-color divergent colormap (see Fig. 5.7(bottom left)).

This viewpoint allows us to make several interesting observations. The $x$ axis barchart for FBDR shows several bars of similar length to the longest bar (*acidity*). This means either that all these variables are strongly correlated, or that FBDR has problems in projecting these variables into separate areas of the 3D target space. If we, however, look at the LAMP projection's $x$ barchart, we see that variable 1 (*acidity*) is not strongly correlated with any of the other variables (the barchart shows a rapid decrease in bar lengths from variable 1 onwards). As we have seen that the projection errors are roughly equal in the three cases, it means that LAMP achieves a (much) better separation of unrelated variables, and the fact that these look correlated in FBDR is just an artifact of FBDR. As such, we can conclude that LAMP is a better projection than FBDR. Comparing ISOMAP with LAMP, using a similar reasoning, yields a similar conclusion – in particular, LAMP is better than ISOMAP since it has a similar error *and* it creates straight axes, whereas ISOMAP creates a curved *alcohol* axis. All in all, we conclude that, for this dataset and related questions, LAMP is a better projection to further use than FBDR or ISOMAP.

### 5.2.2 The Multifield dataset: Explaining projection shapes

This dataset, part of the IEEE Vis 2008 contest, represents the results of a multifield simulation that aims to model the formation and evolution of the early Universe [131]. The dataset contains 200K points, representing various locations in space, and attributed with 10 variables, that describe the points' matter density, temperature, and concentrations of 8 chemical species. While the full dataset is time-dependent, we chose to examine a single frame or time-step, due to the fact that the projection techniques we are aware of cannot readily handle time-dependent data.

Figure 5.8 shows various viewpoints of the dataset, projected into 3D using LAMP (which, as we have seen in Chapter 3 and earlier in this chapter, has overall good qualities). The first observation we can make, is that the projection appears to look like a saddle-shaped manifold. This is an interesting finding, as it means that the involved 10 dimensions strongly constrain each other in some way.

To understand the above saddle shape, and also how variables relate to each other to create it, we examine next the biplot axes. By freely rotating the projection in 3D, we find out that the spread of variable 7 is the largest of the 10 variables. As such, this variable is, arguably, important in explaining the resulting shape. To better view this relation, we align axis 7 with the $y$ screen axis (Fig. 5.8a). This shows us, quite clearly, that the concavity in the saddle-shape is explained *almost* entirely by variable 7, since its axis is perpendicular to it, and since all other 9 biplot axes appear to be located in a plane orthogonal to axis 7. This finding is confirmed by the $y$ axis legend, where we see that all variables show very little correlation with variable 7.

To further explain the saddle-shape, we should, thus, study its variation in directions orthogonal to variable 7. Looking at the $x$ axis legend, we see that variable 5 has a high

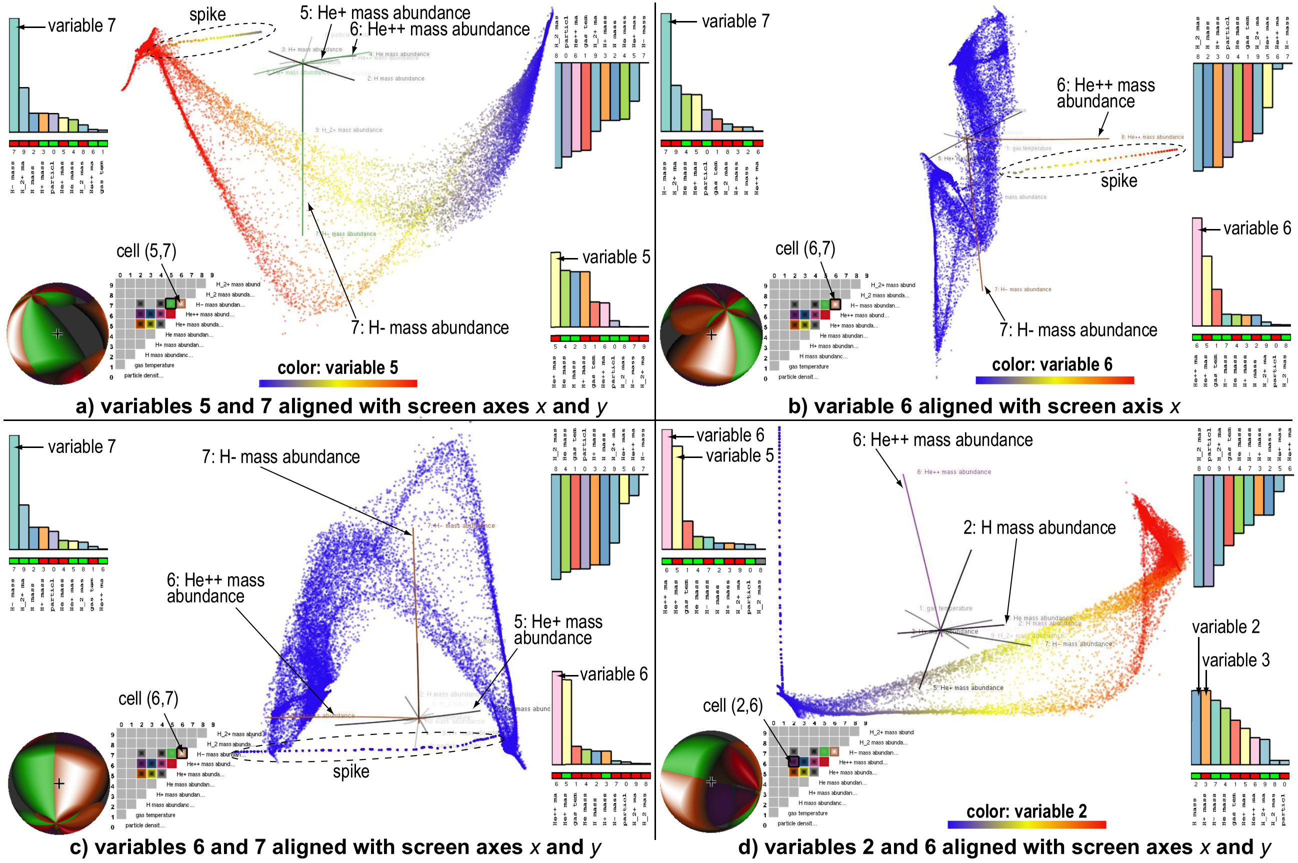


Figure 5.8: 3D LAMP saddle-shaped projection of 10-variate multifield simulation dataset (see Sec. 5.2.2).

bar. Further aligning variable 5 with the $x$ axis, by shift-clicking its bar in the $x$ legend, yields the image in Fig. 5.8) a. We see here, indeed, that the points' spread orthogonal to variable 7 is caused mainly by variable 5. Further on, we see that the icon below variable 5 is red, indicating that it is large to the left, and small to the right, of the image. Color-coding the projection by variable 5 confirms this finding – in other words, the horizontal spread of the saddle is mainly caused by variable 5.

Besides the overall saddle-shape, we also see a 'spike'-like feature in the projection (Fig. 5.8)a, top-left). By slightly turning the viewpoint around its current position, we see that this spike aligns well with both axes 5 and 6. To see which of these two best explains the spike, we align the $x$ axis, in turn, with variables 5 and 6. Comparing the plots, we see that the spike aligns better with variable 6. The plot for this configuration is shown in Fig. 5.8b. To validate the finding, we color points now by variable 6. The result now nicely shows that the spike can be (almost perfectly) explained as containing points having medium-to-large values of variable 6, while the remainder of the projection contains low values for this variable.

The findings so far indicate that variables 6 and 7 are very important in explaining the projection shape. These variables are also quite independent on each other, as shown by both the orthogonality of their biplot axes, but also by the large brown zone on the viewpoint sphere in Fig. 5.8b, which corresponds to the highlighted cell $(6.7)$ in the matrix legend. To study the remaining variables, we choose to align the already studied variables 6 and 7 with the $x$ and $y$ screen axes, respectively. We now get a good view of both the saddle shape and the spike feature (Fig. 5.8c). Additionally, we see here that biplot axes 6 and 6 are almost parallel, so the corresponding variables are strongly correlated. Note that the same finding can be got (albeit with more effort) using the viewpoint legend in this figure: The current viewpoint (cross cursor) best shows, by construction, variables 6 and 7, and is located inside a brown zone on the sphere *but* very close to the border with a green zone. Examining the green cell in the matrix legend, we find that it maps variables 5 and 7. We also see that the green-brown border area is quite bright. All in all, this means that there are many viewpoints which best show a scatterplot of variables 5 and 7 *and* these are very close to viewpoints which show a good scatterplot of variables 6 and 7. Putting it all together, it follows that variables 5 and 6 are strongly correlated.

To further explore the saddle shape, we align variable 6 aligned with the $y$ axis, and align variable 2 with the $x$ axis, yielding the result in Fig. 5.8d. The $x$ and $y$ axis legends tell us now that variables 5 and 6, respectively 2 and 3, are highly correlated, since they have nearly equal *and* almost maximal bars. Coloring this view by the value of variable 2 shows that the rightmost upwards-twisted tip of the projection (red) is explained very well by extremal values of variable 2.

All in all, the above exploratory scenario let us discover that the projection has a 2D-manifold-like saddle shape; that a spike outlier is present; and explain both the saddle shape and spike by the variations and/or values of a few variables. These findings are interesting by themselves, so we may claim that our proposed explanatory mechanisms were *effective* with respect to their supporting aims. A separate aspect regards the *efficiency* of these mechanisms. To perform our exploration, we required about 30 seconds of free rotation (to familiarize ourselves with the overall projection shape), followed by several short sequences of a few clicks each (to align the viewpoint with desired axes pairs) and, optionally, to color-code the projection by the values of one of the variables we aligned with. In contrast, a classical 3D projection exploration tool, consisting of a raw point cloud, virtual trackball, and the ability to color-code points by any of the 10 dimensions, would take much more time to arrive at the same results. While we did not perform a formal measurements here, freely rotating the viewpoint with the trackball so as to best see the spike outlier, takes about 2 to 3 minutes. As explained earlier, this takes just two clicks with our method. Once the alignment is achieved, finding which variable best explains the spike would take cycling through color-coding of the projection by the values of all 10 variables, and memorizing the one which produces the strong color gradient visible in Fig. 5.8b, *i.e.*, variable 6. In our case, this iterative color-coding is not needed – the axis alignment in Fig. 5.8b already tells us that the spike is best explained by variable 6; we next use color coding mainly as a check, and to get a more precise understanding of the *values* of variable 6 that explain the spike.

### 5.2.3 The Segmentation dataset: Comparing 2D and 3D projections

The third dataset we use consists of 2300 19-dimensional points. Each point encodes information about a $3 \times 3$ pixel-block which is randomly-selected from 7 manually segmented outdoor images of various types. The pixel block is described in terms of classical image descriptors, or features, used in pattern recognition and image classification. These include various color and luminance statistical metrics, such as mean and standard deviation, horizontal and vertical contrasts (edges), and gradients [116]. The $19^{th}$ attribute describes the label, or class, of the image, and has 7 distinct values, corresponding to the 7 image examples. This dataset is frequently used in applications involving the construction of automated image classifiers that aim to learn the 7 classes from the values of the 18 measured image attributes [95, 141, 138].

To explore this dataset, we project it to three dimensions using LAMP (Fig. 5.9). Next, we color points by the value of the categorical label attribute (which is not used in the projection). Our aim is to see, on the one hand, how the measured variables correlate (or not) with each other, and how they correlate with the class attribute – similarly to the use-case described in Sec. 5.2.1. We start by freely rotating the 3D projection to get an overall impression, as for the example discussed in Sec. 5.2.2, and see that the longest biplot axis corresponds to variable 0 (*region-centroid-col*). As such, we align this variable with the $y$ screen axis, to 'factor it out' from the projection, and see which other variables explain the projection's shape. After alignment, we see that the current viewpoint corresponds to a red cell in the matrix legend (Fig. 5.9a), which maps the variable-pair $(0, 3)$. The large red zone on the viewpoint sphere means that, if we turn the viewpoint even with large amounts from the current position, we still can best see variables 0 and 3. This is not too surprising, given the very large length of axis 0 as compared to *all* other axes. Aside from this, we are next interested to explore variable 3 (*short-line-density*), so we click this cell and obtain the viewpoint which best explains these two variables (0 and 3).

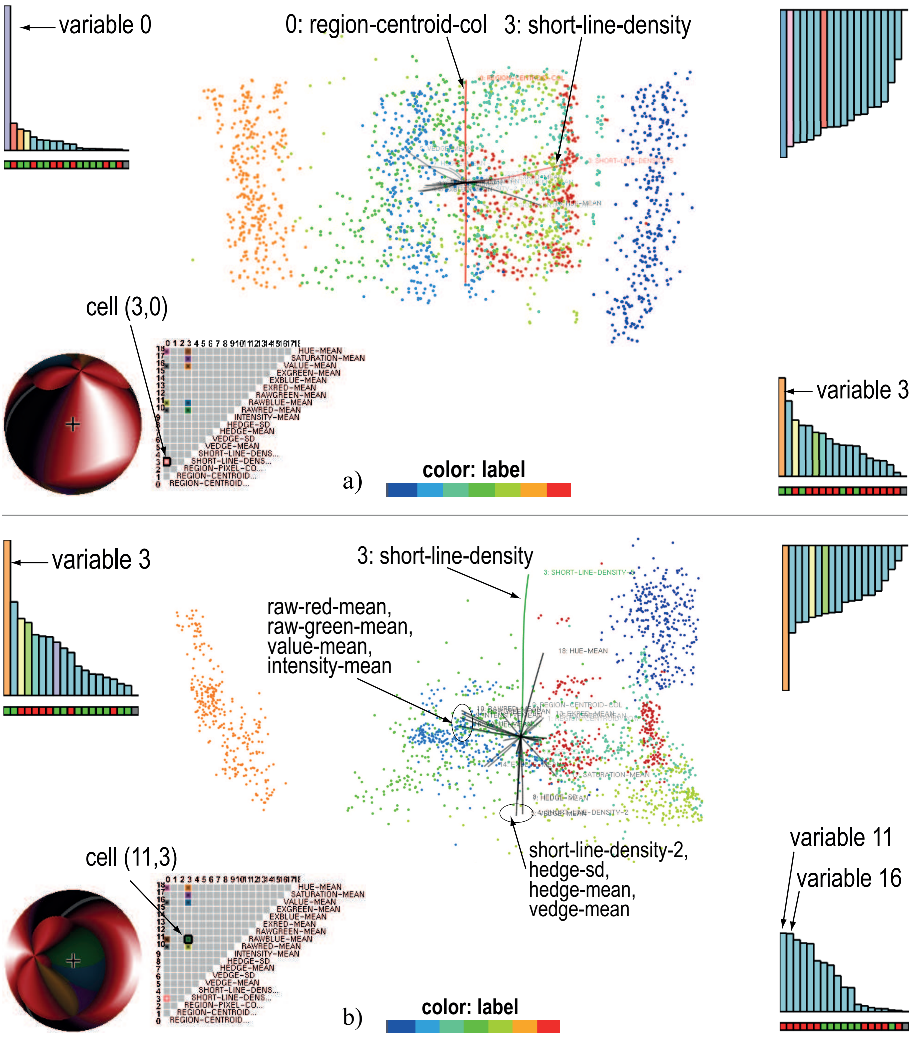


Figure 5.9: Visualization of 19-variate image dataset using 3D projections (a,b). See Sec. 5.2.3.

From the current viewpoint, two useful insights can be drawn. First and foremost, the vertical same-color 'bands' present in the figure show that there is no correlation of specific values of variable 0 with the label values – in other words, any label value occurs equally frequently for any value of variable 0. Secondly, we see that all biplot axes 1-8 are located in a plane roughly perpendicular to axis 0. This plane also contains the directions along which the colors mapping the labels mostly vary. Taken together, all facts above tell us that variable 0 is (a) independent on all other variables, and (b) not correlated with the label values. As such, we can, and actually we should, eliminate variable 0 from the dataset, as it brings no added value for classification.

Given the above, one can proceed further in various ways. Among other options, we consider the following:

1. Construct a new dataset where variable 0 is removed, and repeat the entire exploratory procedure;
2. Use the current 3D projection in such a way that the (uninteresting) effect of variable 0 is removed;
3. Use a 2D projection of the dataset.

While option (1) above is interesting, we decided not to study it further, since removing an actual dimension from the input dataset may, after all, cause unexpected and undesired effects. Also, comparing our results with other techniques which use this same dataset would become more difficult in this case.

To study option (2), we align biplot axis 0, which corresponds to the dimension we found to be uninteresting, with the viewing direction. Further on, we align biplot axis 3, which we already have found as being important, with the screen $y$ axis. The result is shown in Fig. 5.9b. In this figure same-color point clusters emerge more visibly than in Fig. 5.9a, which suggests that it is, indeed, possible to compute the label (class) values based on the 17 remaining dimensions. To refine this insight, we study the biplot axes in Fig. 5.9b. As a first observation, we see several variable correlations, *e.g.* the vertical downward-pointing axes for *short-line-density-2*, *hedge-sd*, *hedge-mean*, and *vedge-mean*). These correlations may not be too surprising, as all these variables describe essentially image edges. Similarly, we see a group of correlated axes pointing horizontally to the left (*raw-red-mean raw-green-mean*, *value-mean*, and *intensity-mean*). As for the former group, these correlations are not that surprising, as all these variables essentially describe color averages.

The above correlations, although explainable once we have found them, indicate that there are a number of possibly redundant variables, or features, in our dataset. This is a quite well known phenomenon in feature extraction for image classification – adding several types of edge-detector, gradient, or statistical features to a multidimensional dataset does not necessarily increase the true dimensionality of the data, and thus of the performance of classifiers built atop of it, as such features are often strongly correlated. However, in the same time we see that certain clusters can be easily explained by the existing variables. For instance, the isolated orange cluster left in Fig. 5.9b can be easily explained in terms of highly-saturated colors. This can be a first step into building a (simple but robust) classifier that first isolates images corresponding to the 'orange' class from all others.

To study option (3), we project the data into 2D, using LAMP, and color points by label values (Fig. 5.10a). Interestingly, the obtained pattern of same-color point clusters is relatively similar to those obtained in the specific 2D view of our 3D projection shown in Fig. 5.9b. This raises the question of which, of these two techniques (selecting a suitable view of a 3D projection *vs* projecting the dataset directly to 2D) is the best one.

To study this, we compute and display the the aggregated normalized projection errors $e_i^m$ defined earlier in Eqn. 5.1. Figures 5.10b and 5.10c show the errors $e_i^3$ and $e_i^2$ for the 3D and 2D projections in Figs. 5.9b and 5.10a respectively. In both cases, we color map errors using a blue-yellow-red divergent colormap. By comparing the two

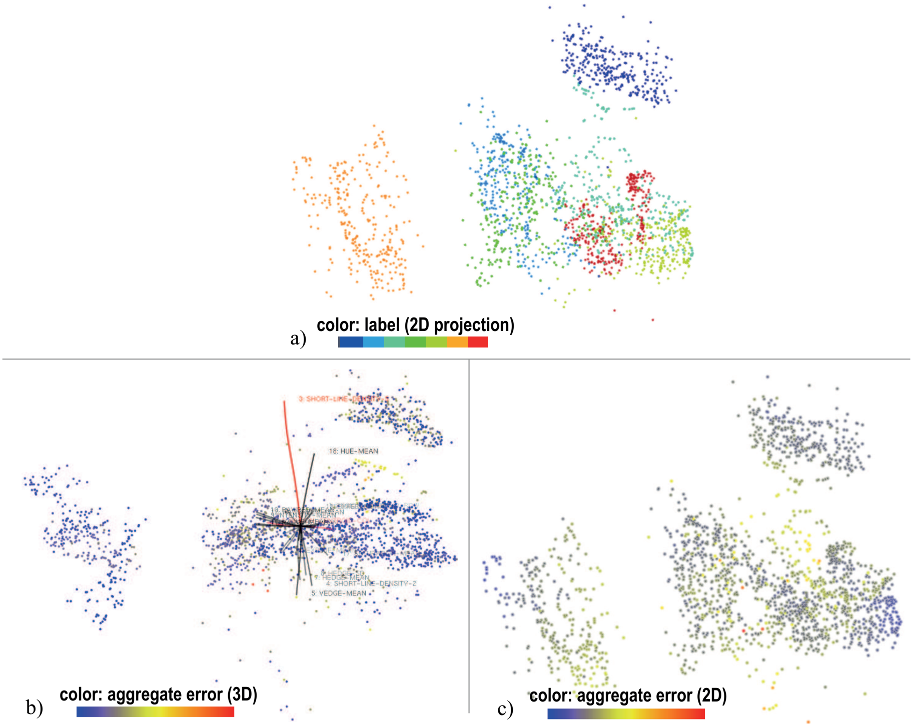


Figure 5.10: 2D LAMP projection of a 19-dimensional dataset showing point labels (a) and errors (c). 3D LAMP projection of the same dataset showing errors (b). See Sec. 5.2.3.

images, we immediately see that $e_i^3$ is overall lower than $e_i^2$. This was not unexpected – as discussed earlier, it is easier to preserve distances when mapping to a higher target dimension. Separately, we see that the distribution of errors over points is quite similar in the two projections – that is, there are overall just a few very high-error points, and there is a limited spatial variation in error magnitude between neighboring points. From the above, we can conclude that using a suitable viewpoint of our 3D LAMP projection is, error-wise, better than using a direct LAMP projection to 2D. This conclusion can be explained, among others, by the fact that the 2D view of the 3D projection in Fig. 5.10b places the large variation of variable 0 along a dimension which is orthogonal to the view plane. In contrast, in the 2D projection case (Fig. 5.10c), this large variation has to share the same (limited) 2D projection space which is used by the other 17 variables. All in all, visualizing the 3D LAMP projection from the viewing direction that cancels out the effect of variable 0 is conceptually identical to creating a 2D projection that uses only variables $1 \ldots 18$, *i.e.*, identical to our option (1) outlined above. As this does not explicitly alter the dataset by removing a dimension, and also creates a projection having lower error than directly projecting all data to 2D, we conclude that option (2) – visualizing a 3D projection from a suitably chosen viewpoint – is the optimal one in this case.

### 5.2.4 The Software dataset: Finding meaningful clusters

For our fourth example, we consider a dataset describing a corpus of open-source software projects written in C [121]. For each project, 11 software quality metrics were computed, by statically analyzing their source code files, and averaging the results for the entire project. The provided database contains 6733 projects (observations), each having 11 attributes (code quality metrics). A separate $12^{th}$ attribute measures the number of times each project was downloaded. Similar to [121], we are interested in finding correlations between the metrics themselves and/or the metrics and the download count values.

As in the previous examples, we start by freely examining a 3D projection of our data, computed using LAMP (Fig. 5.11). Here, in contrast to the previous examples, we continue our exploration by using the matrix legend, rather than the biplot axes and axis legends. We note that this is not a mandatory choice, but just a means to show alternative investigation paths that our interactive tools support.

In this matrix legend, we notice an outstanding bright green cell, indicating that there is a pair of variables that admits to be very well examined from certain viewpoints. We select this variable-pair, *i.e.* variable 2 (*ln-cof*, or average coupling-factor, *i.e.* the number of inter-file function-calls [107]) and variable 7 (*ln-sum-tloc*, or total number of lines-of-code), and its best viewpoint by clicking on this green cell, and obtain the view in Fig. 5.11a. We next refine this view by aligning biplot axis 2 with screen $x$ axis. In the resulting image (Fig. 5.11a), two large point clusters $A$ and $B$ emerge. The spread in both clusters appears to be perpendicular to axis 2 (*ln-cof*). In other words, points in $A$ and $B$ seem to be distinguishable by the values of *ln-cof*. To further see this, we color-code all points by this variable, and see, indeed, that points in $A$ contain low *ln-cof* values. Upon examination by brushing, we found out that these low-coupling systems are mainly libraries. In contrast, points in $B$ contain average-coupling systems, such as standalone programs (applications). In the same view, we notice a third, separated, cluster $C$ that mainly corresponds to very high coupling systems. However, to better characterize $C$, we now see that its shape is orthogonal to axis 7 (*ln-sum-tloc*), and it is positioned close to one end of this axis. Hence, variable 7 can be useful (aside of variable 2) in explaining cluster $C$. To get more insight, we color next all points by variable 7 (Fig. 5.11c): As observed earlier, we confirm the fact that points in $C$ have similar *and* extremal values of variable 7 – specifically, they have very low values of this variable, whereas both clusters $A$ and $B$ have relatively high values for variable 7. Since variable 7 denotes the total size of a system (*ln-sum-tloc*), it means that $C$ can be explained as containing small systems, whereas $A$ and $B$ contain large systems. Putting it all together with the earlier observations, we explain $C$ as containing small applications; $A$ as containing large libraries; and $B$ as containing large applications, respectively.

Let us now consider the same question here as for the imaging dataset discussed in Sec. 5.2.3: Would we obtain the same insight (splitting of the software corpus into three types of systems) if we used a 2D projection? We start exploring this question in the same way as in Sec. 5.2.3 – namely, we compute a 2D LAMP projection of the same dataset, and next compare the aggregated projection errors $e_i^3$ (Fig. 5.12a) and $e_i^2$ (Fig. 5.12b). We see that both $e_i^3$ and $e_i^2$ are uniformly distributed over their 3D,

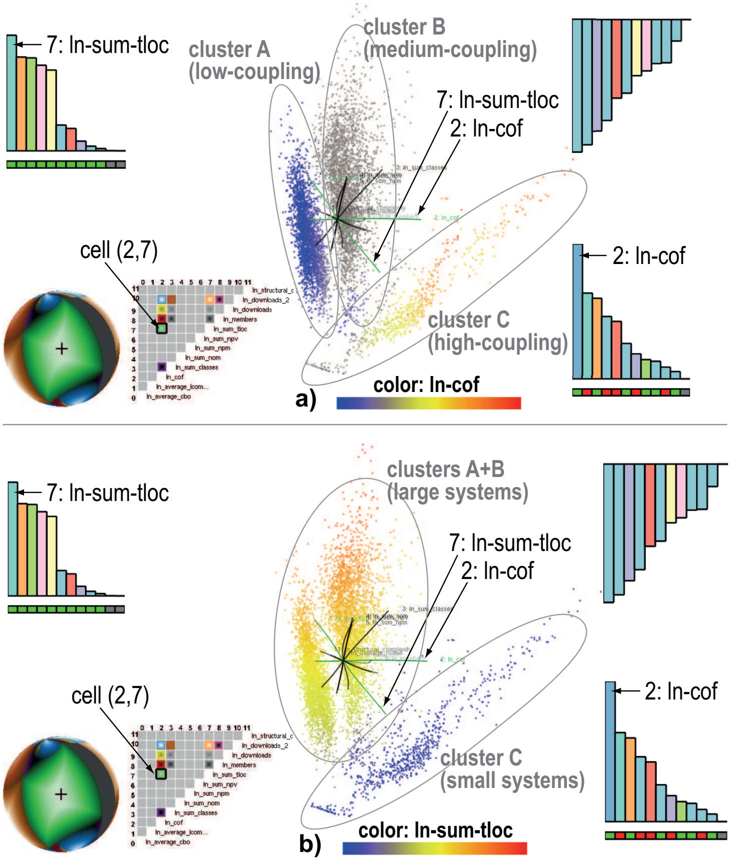


Figure 5.11: Visualization of 12-variate software metrics dataset using 3D LAMP. See Sec. 5.2.4.

respectively 2D, projections, and that $e_i^3$ is in general smaller than $e_i^2$. These insights are very similar to the ones obtained for our imaging dataset.

A main difference with the earlier imaging dataset use-case is that the *shapes* of the 2D and 3D projections are quite different for our software dataset. Indeed, the 3D projection, viewed from the viewpoint discussed earlier on, shows three clusters, which, as we have seen, can be explained mainly by two variables (*ln-cof* and *ln-sum-tloc*, as discussed). In contrast, the 2D projection shown in (Fig. 5.12b) depicts only two clearly separated clusters $A'$ and $B'$. To explain these, we brush their points and iteratively color the 2D projection by the values of the variables (since we use a standard 2D LAMP implementation which does not feature any explanatory tools). This way, we found that $A'$ contains a mix of large libraries *and* applications (thus, an union of $A$ and $B$); and $B'$ contains small systems (thus, roughly corresponds to $C$). As such, our software dataset studied here is, in contrast to the imaging dataset, an example where a 3D projection is truly needed to let us discover the separation of the point-set into three distinct classes (using just a 2D projection would only elicit two such classes, thus, a coarser-grained explanation of the data).

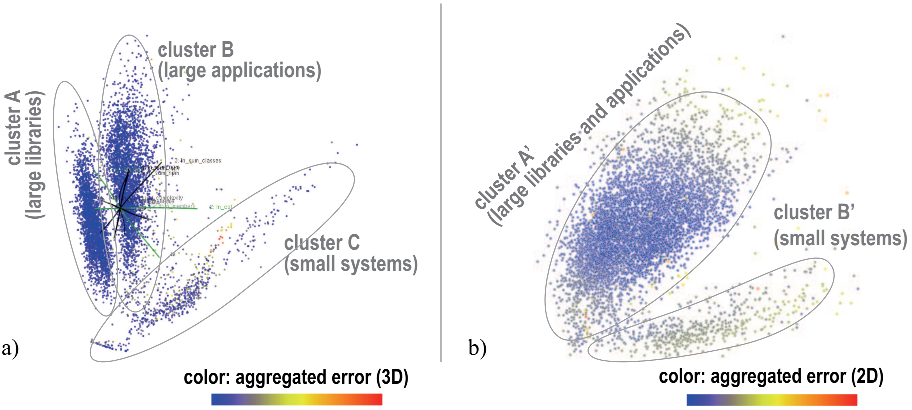


Figure 5.12: Software dataset: Comparing a selected view of a 3D LAMP projection, and its aggregated error (a), with a 2D LAMP projection and its corresponding error (b). See Sec. 5.2.4.

## 5.3 Discussion

Our proposed explanatory techniques for 3D projections raise several discussion points, as follows.

**Added value:** The explanatory techniques proposed in this chapter are effective to show the presence of strongly (inversely) correlated dimensions, independent dimensions, and to explain groups of closely-projected points (clusters) in terms of specific value ranges of specific dimensions. However, it should be noted that the last feature is only effective if such clusters are already present in the underlying 3D projection. If the projection fails to identify such clusters, or if such clusters actually do not exist in the original high-dimensional space, the added-value of our techniques will be limited to the former use-cases listed above. To a larger extent, this limitation also applies to finding correlations (or the lack thereof) between variables. One way to test the effectiveness of using our techniques is to study the projection errors (Chapter 3) prior to their application: If such errors are small, we believe that it is very likely that our techniques will next help with the above-mentioned explanatory tasks. If the projection exhibits large errors, the exploring its parameter space to improve it (see also Chapter 3) should be done prior to exploring the projection.

**3D *vs* 2D context:** As outlined by our examples presented in Sec. 5.2, our techniques focus on explaining 3D projections. In contrast to 2D projections, the user's choice of viewpoint for 3D projections is crucial with respect to the obtained insights [139, 163]. To explain this further, we can consider a view of a 3D projection as being the composition of a $n$D-to-3D projection ($P_{n3}$) with a 3D-to-2D typical orthographic or perspective projection ($P_{32}$). Denote the projection of the same dataset from $n$D to 2D by $P_{n2}$. As discussed at various points, the error of $P_{n3}$ is in general lower than the error of the corresponding $P_{n2}$ (all other factors such as projection method, parameters, and input dataset being kept the same). However, the total error that our 3D pipeline creates, also contains the error introduced by $P_{32}$. Hence, to minimize this total error, we aim to minimize the error of $P_{32}$. Our exploratory and explanatory tools attempt to do this *selectively* – that is, to minimize the error of $P_{32}$ for specific sets of points, or sets of dimensions. This allows one to obtain a view in which such specific subsets of interest of the input data are projected well to the final 2D image, whereas other subsets of the data may (still) contain large errors. The key added value of our interactive techniques resides precisely in being able to easily specify on-the-fly which these subsets are. For instance, we can change the viewpoint to optimize the comparison of a selected pair (or set) of variables. The same can be said concerning occlusion, a factor which is known to adversely affect the exploration of 3D scatterplots: It is true that 3D projections are liable to occlusion. However, one has still the chance of changing the viewpoint so as to decrease occlusion selectively for given subsets of points of interest. In contrast, the same is not true for 2D projections: Overlaps also occur in such projections, and reducing them cannot be done by other means than re-computing the entire projection.

**Generality:** The proposed mechanisms can directly handle any dimensionality-reduction technique, whether linear or not, global or local, white-box or black-box, and using the actual high-dimensional point coordinates, or alternatively a distance matrix (MDS-like methods). In all cases, we do not need any access to, or information about, the projection algorithm's internals. This cannot be said about other explanatory mechanisms for projections [71, 1, 132, 24], which make explicit use of the projection algorithm's internals (typically, SVD or PCA). As such, we were able to easily use our explanatory tools with projections as diverse as LAMP, ISOMAP, and FBDR (demonstrated here) and also LSP [139] and PLMP [141] (omitted here for brevity). Applying our techniques to additional projection methods, such as *e.g.* the very popular t-SNE method [192, 194] can be immediately done.

**Scalability:** Computationally speaking, our techniques scale linearly with the number of dimensions, which makes them applicable to real-time data exploration of very large projections. Visually speaking, our same techniques scale quite well up to roughly $n_{max} = 20$ variables, which is in line with the observed behavior of other multivariate visualization techniques [132, 50, 24, 31]. For datasets having many more dimensions, we choose to limit insights to the most important (salient) dimensions that can be explored from a given viewpoint (axis biplots, axis legends) and also the globally most salient dimension-pairs (viewpoint legend). While this can be seen as a limitation in the view of existence of datasets having hundreds up to thousands of dimensions, we believe that aiming to display (information about) *all* these dimensions can be a too ambitious, and possibly even impractical, endeavor. For such datasets, the typical way to handle them is to first automatically aggregate or cluster dimensions by data mining and machine learning techniques, and visually explore the reduced-dimensionality result.

**Comparison:** Our techniques bear several similarities with existing work in exploring multidimensional datasets. Apart from the already explained similarities with biplot axes and axis legends [1, 24], we note here also ‘rolling the dice’ (RTD) [50]. Our axis-alignment (Sec. 5.1.3) and viewpoint-selection mechanisms (Sec. 5.1.6) are conceptually similar, in purpose, to the cells of a scatterplot matrix, in the sense that they display selected ‘interesting’ variable-pairs. This also bears a resemblance to scagnostics techniques which aim at pre-selecting interesting variables to display next [207]. Yet, differences exist. Most existing techniques display such interesting variable-pairs as classical Cartesian scatterplots, whereas we define them as viewpoints on a given 3D projection. One disadvantage of our approach is that, unlike Cartesian scatterplots, we are limited, in terms of quality and accuracy, by the configurations encoded by the 3D projection. In contrast, one advantage is that any of our viewpoints displays information about more than two (in theory, all) dimensions, while classical scatterplots are limited to two up to three dimensions.

**Technical details:** Several of our design decisions, such as the precise choices made for colormaps and color transfer functions are, of course, open to customization and/or improvement. For instance, one can customize the categorical colormap used in the axis legends to mark specific variables of interest, which one needs to pay particular attention during the analysis, with salient colors or colors having an application-specific semantics. Alternatively, one could select an axis legend, color map its bars using a sequential or ordinal colormap, and next compare this legend color-wise with the other two axis legends to reason about variable correlation or orthogonality. Yet other alternatives may exist for specific user groups and work domains.

**Limitations:** From a technical perspective, our visualizations are limited in terms of number of observations they can display without causing excessive occlusion (a limitation shared by any 3D scatterplot technique) and the number of dimensions that can be simultaneously explained. Separately, it can be argued that the proposed tools only address a limited subset of the questions and tasks related to multidimensional data exploration. For instance, we do not support the explicit identification of well-defined point clusters and the explanation of such clusters in terms of dimension values and/or dimension correlations, but only provide a global explanation of the entire dataset. Separately, the viewpoint legend focuses mainly on showing the best-visible pairs of independent variables. It can be argued that extending this legend to show (viewpoints from which we can analyze) pairs of strongly correlated variables, or extending both above scenarios to sets of more than two variables, is an useful addition.

In terms of user effort, our techniques cannot, and do not aim to, fully replace interactive trial-and-error exploration, as compared *e.g.* to certain scagnostics-based techniques. Rather, our aim is to *support* interactive exploration and make it reach a given goal more efficiently. Examples hereof include quickly finding suitable viewpoints for certain tasks, understanding what a viewpoint tells, but also providing visual feedback as one freely rotates the projection so as to pre-attentively guide the user towards specific navigation paths.

Separately, we note that the discussion of certain advantages of 3D projections with respect to their 2D counterparts (Sec. 5.2) should not be generalized to imply that 3D projections are *always* better, for *all* exploration tasks and datasets, than 2D projections. Rather, the aim of the respective discussion is to show that there are situations in which 3D projections can be more effective than 2D projections, when complemented by suitable exploratory tools.

## 5.4 Conclusion

In this chapter, we have presented several interactive visualization techniques aimed at assisting users when exploring three-dimensional projections of high-dimensional data. The core aim of these techniques is to add back to a 3D projection information in terms of the identities, ranges, values, and (cor)relations of the original high-dimensional variables, specifically when such information has been lost during the projection itself. The added information helps in understanding patterns and shapes created by point groups in the 3D projection in terms of the original variables; identify groups of strongly correlated, or independent, variables; identify limitations of the projection; and, last but not least, help the user in understanding what a given viewpoint of a 3D projection does show, and how to suitably choose such viewpoints for specific tasks. Our techniques are computationally scalable, simple to implement, and can be directly added to any type of dimensionality-reduction method, without any specific constraints. From a user perspective, our methods complement, rather than replace, classical exploratory techniques such as brushing, selection, and color mapping.

As a consequence of having these techniques, we have shown that 3D projections can be used to obtain more precise, and sometimes more detailed, insights as compared to their 2D counterparts. For these cases, we have also shown that 3D projections minimize the distance-related errors implied by reducing dimensionality, as compared to the same 2D counterparts; and that exploring these 3D projections does not bring considerable additional costs as compared to exploring 2D projections. Taken together, the above insights indicate to us that 3D projections should not be dismissed as an alternative to 2D projections for the task of high-dimensional data exploration.

The first contribution of this chapter – explaining 3D projections in terms of biplot axes and axis legends – is a first step towards making scatterplots produced by projection techniques interpretable in a more intuitive way by a wide range of users. In the next chapter, we refine this idea of explaining projections by proposing a set of alternative visual techniques that complement the work presented here.

## Contributions

The work presented in this chapter is based on the article *Explaining Three-dimensional Dimensionality Reduction Plots* (D. Coimbra, R. Martins, T. A. T. Neves, A. C. Telea, F. V. Paulovich), Information Visualization Journal, SAGE Publications, September 2015

(DOI:10.1177/1473871615600010). The first two co-authors have had equal major contributions to the publication, and should be seen as joint first authors. Furthermore, specific parts of the work can be assigned to R. Martins, as follows: the analysis of three-dimensional projection errors and the extension thereof to viewpoint-dependent errors (Sec. 5.1); the elicitation and testing of the specific use-cases for the viewpoint legend widget (Sec. 5.1.6); the analysis of the *wine* dataset and usage of projection errors to derive additional insight in comparing projections (Sec. 5.2.1); and the selection, analysis, and interpretation of the *software* dataset (Sec. 5.2.4).

## Chapter 6

# Local Explanation of Multidimensional Projections

In the previous chapters, we have explored several ways for explaining multidimensional projections. Chapters 3 and 4 show how projection errors related to the preservation of inter-point distances, respectively point neighborhoods, can be visually encoded so that users get a detailed insight into where, how large, and which types of errors occur in such projections. Chapter 5 shows how the 3D low-dimensional projection space can be annotated with axis legends to explain the directions of variation of the original high-dimensional attributes, by adapting and extending existing explanatory techniques for 2D projections.

While contributing in different ways to explaining projections, the above techniques fall short of explaining the reason why observations are placed close to each other in projections: The techniques presented in Chapters 3 and 4, while providing local insights in a projection, focus only on showing the *errors* caused by the underlying projection method. As such, these techniques are mainly useful to decide which projection areas are reliable in the sense of faithfully encoding the high-dimensional data structure, and which areas contain significant errors. As explained in Chapter 3, a typical workflow involving error analysis focuses mainly on limiting the subsequent data analysis to low-projection areas. For such areas, we can next use the explanatory techniques presented in Chapter 5, *i.e.* biplot axes and axis legends, to get an overall impression of how the high-dimensional variables spread across the projection space (which can be two- or three-dimensional). These techniques are useful in understanding *global* projection features, such as correlations or orthogonality relations between variables; or explaining outliers which are well aligned along high-dimensional variables. However, these techniques fall short in understanding *local* projection features, such as small groups of closely-placed observations.

In this chapter, we address the goal of explaining multidimensional projections from the perspective of local structures apparent in the projection. In detail, we enrich 2D scatterplots created by multidimensional projection techniques, by visually highlighting the key dimensions, or attributes, that make closely-projected points similar. These explanations are different from those proposed in Chapter 5: While the biplot axes and axis legends show the directions of maximal variation of high-dimensional attributes, the explanations proposed in this chapter show the attributes which determine the appearance of each local neighborhood of points in the projection – hence, the name 'local explanation' used for these techniques. We compute such explanatory visuals over all point neighborhoods of a 2D projection and render them next by image-based techniques, similar to the ones presented in Chapters 3 and 4. Our technique implicitly partitions the projection space into compact areas that are next provided with simple explanations in terms of variables. The technique is simple to implement and independent on the (2D) projection technique being used.

The structure of this chapter is as follows. Section 6.1 summarizes related work targeting the explanation of multidimensional projections in terms of attributes. Section 6.2 introduces the proposed visual explanatory techniques. Section 6.3 presents applications of our techniques on several real-world multidimensional datasets. Section 6.4 relates our technique to all other visual explanation techniques proposed in this thesis. Section 6.5 concludes the chapter.

## 6.1 Related Work

Multidimensional projections are effective tools in showing groups of similar observations present in large datasets where each observation is described by many dimensions or attributes. However, in practice, such projections are very challenging to interpret: While standard projections show, indeed, which observations are similar, they fail to explain *why* observations are similar. This type of information is crucial in further using the insight regarding the similarity of groups of observations.

The need to explain projections has been long recognized in information visualization and visual analytics. The most natural way to explain such projections is in terms of the original high-dimensional variables, which encode domain-specific knowledge, and are arguably the most intuitive way for users to understand them. To this end, several explanatory mechanisms have been proposed. As outlined in Section 2.5, such mechanisms can be classified into several types, as follows.

*Interactive approaches* show information on-demand atop the projected observations, such as color-coded values of a user-selected attribute or user-selected values of attributes mapped to additional points in the projection [51]. More advanced methods target the finding of interesting feature subspaces to display the data within, in a scagnostics-like fashion [177]; feature selection for exploration via the visualization of aggregated feature relevance data [105]; and the interactive selection of feature subspaces by the visual analysis of plots combining observations and attributes [211]. However, such interactive methods require a significant effort from their end users. Biplots and biplot axes show the directions of variation of all attributes atop the projected observations [1, 71, 69, 189]. However, such methods still keep a certain separation between observations and attributes, in the sense that they do not explicitly show why observations are placed at their respective positions in the projection. Axis legends show the attributes that have the highest variance along the projection-space axes, and are applicable both for 2D projections [24, 132] and 3D projections (as shown by our work presented in Chapter 5). However, such methods only explain the projection-space axes, and do not explicitly mark each point cluster in the projection with the attributes that determined its appearance.

A different way to explain multidimensional projections is offered by *clustering* methods. Such methods explicitly partition the projection into clusters of close points. Assuming, next, that the technique used to generate the projection places indeed points which are similar in the input high-dimensional space close to each other, an explanation is synthesized for each such cluster, *e.g.* in terms of one or a few representative samples that best characterize the data variation within a cluster, or in terms of the dimensions that have the lowest variance over the samples in a cluster [175, 130, 101]. This provides an effective and visually scalable summarization of the projection in terms of a few representatives or attribute names. However, clustering methods are challenged by the need to explicitly decide where to draw borders between sets of observations. If such borders are not constructed appropriately, the resulting visualization may convey wrong insights. Additionally, clustering implies a hard partition of the projection into distinct regions which are next separately explained. This works well for projections consisting of several compact groups of points separated by large amounts of white space, but has problems for projections which exhibit a fuzzier, more subtle, variation between close points.

In machine learning applications, *scoring metrics* are proposed to determine which attributes, also called features, determine a user-selected number of high-dimensional observations to be close together (so called compactness metrics) or, alternatively, which attributes make two or more groups of observations be far away from each other (so called segregation metrics) [72, 64]. In spirit, such methods are close to the aim of the work presented next in this chapter – that is, determine, or rank, the dimensions which are mainly responsible for the appearance of groups of close observations in projections. Yet, a key difference exists between the typical use of dimension-scoring metrics in machine learning and the use of dimension-ranking metrics in our own work:

- Dimension-scoring (in the first context) is typically done to explain why an *explicitly selected* group of observations is different than another group of observations. Explicit selection is done either by the user, or by using the value of a so-called class attribute, in typical classifier design;
- Dimension-ranking (in our own context) will be done to explain why an *implicitly determined* group of close observations is different from all other observations in the dataset. There is no user-based or attribute-based explicit delimitation of such groups. Rather, they are formed implicitly by the fact that our method yields the same explanation for neighboring points.

Moreover, feature scoring techniques, while popular in machine learning, have been quite sparsely used for the visual explanation of multidimensional projections.

## 6.2 Method

### 6.2.1 Concept

Consider a high-dimensional dataset $D^n \subset \mathbb{R}^n$, and a projection function, or method, $P : \mathcal{P}(\mathbb{R}^n) \rightarrow \mathcal{P}(\mathbb{R}^m)$, where $m \ll n$, and $\mathcal{P}$ denotes the power set operator. In our work next, we shall fix $m = 2$, *i.e.*, use two-dimensional projections. We call the value $D^m = P(D^n)$ the projection of $D^n$ into a low-dimensional dataset $D^m \subset \mathbb{R}^m$, At the core of our proposal is the attempt to answer the following question:

*Given a group of close neighbor-points $\nu^m \subset D^m$, which dimensions $V \subset \{1, \ldots, m\}$ of the original high-dimensional data are the most responsible ones for the fact that the* *points in $\nu^m$ are close to each other?*

Let us further refine this question. Assuming the projection function $P$ is a 'well behaved' one, it will aim to preserve distances (like *e.g.* most multidimensional scaling techniques), or alternatively preserving neighborhoods (like *e.g.* t-SNE [194]) when projecting points from $\mathbb{R}^n$ to $\mathbb{R}^m$. Consider now a small point-neighborhood $\nu^m \subset D^m$, and denote by $\nu^n$ the high-dimensional points that project there, *i.e.*, $P(\nu^n) = \nu^m$. If, indeed, $P$ preserves distances and/or neighborhoods, as it is expected from a good-quality projection, then the points $\nu^m$ also form a (close) neighborhood.

The key question next is: Can we determine some particular statistical aspect of the data values of the points in $\nu^n$ which explains why those points are close to each other in $D^n$, *i.e.*, form a neighborhood? If so, then the respective aspect is *also* the reason why $P$ has created $\nu^m$ in $D^m$. Hence, by annotating $\nu^m$ with the respective aspect, when visualizing the projection $D^m$, we explain to users why the respective neighborhood $\nu^m$ has been formed by the projection.

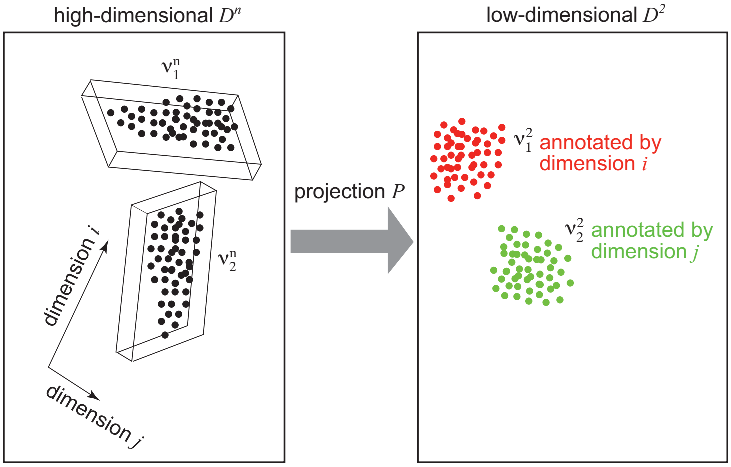


Figure 6.1: Annotating point-neighborhoods by the dimensions that best explain their existence in the low-dimensional projection space (right). For the high-dimensional neighborhoods (left), their bounding-boxes are outlined.

Figure 6.1 illustrates the idea. In this example, we consider a dataset $D^n$ which consists of two well-separated point clusters $\nu_1^n$ and $\nu_2^n$. Furthermore, let us assume, for the sake of explanation, that the points in $\nu_1^n$ are close because of their strong similarity with respect to some dimension $i$ of the $n$ dimensions of $D^n$; and points in $\nu_2^n$ are close because of their strong similarity with respect to some other dimension $j$. This is shown in the figure by the depiction of the axis-aligned bounding boxes of $\nu_1^n$ and $\nu_2^n$. Assuming that our projection $P$ preserves distances well, the corresponding 2D neighborhoods $\nu_2^1$ and $\nu_2^2$ will also show up as well-separated groups of points in $D^2$. If we can detect this fact, *i.e.*, we can find the identities of the dimensions $i$ and $j$, we can next annotate the corresponding neighborhoods $\nu_1^2$ and $\nu_2^2$ with the explanations $V_1 = \{i\}$ and $V_2 = \{j\}$, respectively. For example, we can color-code the set $V$, using *e.g.* red to map dimension $i$ and green to map dimension $j$, and color map the projected points $D^2$ by this color coding. The resulting visualization (Fig. 6.1 right) then easily explains to the user that the point-cluster $\nu_1^2$ is mainly formed because the respective high-dimensional points ($\nu_1^n$) are close because of dimension $i$, whereas the point-cluster $\nu_2^2$ is mainly formed because the respective high-dimensional points ($\nu_2^n$) are close because of dimension $j$, respectively.

### 6.2.2 Ranking the dimensions

Let us now refine the above idea. We start by defining our two-dimensional neighborhoods $\nu_i^2$ so as to be centered at each projected point $\mathbf{q}_i \in D^2$, as $\nu_i^2 = \{\mathbf{q} \in D^2 | \|\mathbf{q} - \mathbf{q}_i\| \leqslant \rho\}$, where $\rho$ is a (small) radius, or neighborhood size. Given such any neighborhood $\nu_i^2$, we immediately find the neighborhood $\nu_i^n$ of the $n$-dimensional point $\mathbf{p}_i$ that maps, by projection, to $\mathbf{q}_i$, simply by point correspondence. Next, as indicated above, we analyze the distribution of points in $\nu_i^n$ to compute a so-called *ranking*, or ordering, of the $n$ dimensions of our dataset. We denote this ranking by the tuple, $\mu_i = (\mu_i^1, \ldots \mu_i^n) \in \mathbb{R}^n$. Here, $\mu_i^j \in \{1, \ldots, n\}$, $1 \leqslant j \leqslant n$, denotes the fact that dimension $\mu_i^j$ has rank $j$ for the neighborhood centered at point $i$. In other words, $mu_i$ is a permutation of the set $\{1, \ldots, n\}$. Additionally, a low-rank dimension, *i.e.* a value $\mu_i^j$ for a low $j$, is more important in the explanation of $\nu_i^2$ than a high-rank dimension. The remainder of this section introduces two ways to compute such rankings.

**Euclidean ranking:** The intuition behind this ranking method can be found if we consider the squared Euclidean distance between two points $\mathbf{p}$ and $\mathbf{r}$

$$\|\mathbf{p} - \mathbf{q}\|^2 = \sum_{j=1}^{n} |\mathbf{p}^j - \mathbf{r}^j|^2, \tag{6.1}$$

where $\mathbf{x}^j$ denotes the $j^{th}$ dimension of a point $\mathbf{x}$. This distance essentially explains the similarity of the two high-dimensional observations, and is composed out of $n$ contributions of the $n$ dimensions. As such, we can say that a dimension $j$ is contributing the normalized amount

$$lc_{\mathbf{p},\mathbf{r}}^j = \frac{|\mathbf{p}^j - \mathbf{r}^j|^2}{\|\mathbf{p} - \mathbf{r}\|^2}. \tag{6.2}$$

to the (dis)similarity of two high-dimensional points.

We can extend the above idea of dimension-contribution from a point $\mathbf{p}_i$ to the entire neighborhood $\nu_i^n$ centered around it. We define this per-neighborhood dimension-contribution by averaging the per-point dimension-contributions of all the neighborhood's points as

$$\overline{lc_i^j} = \frac{\sum_{\mathbf{r} \in \nu_i^n} lc_{\mathbf{p}_i,\mathbf{r}}^j}{\|\nu_i^n\|}, \tag{6.3}$$

where $\| \cdot \|$ is the set size operator. Intuitively, if $\overline{lc_i^j}$ is low for a dimension $j$, it means that the respective dimension contributes little to the *dissimilarity* of points in the neighborhood $\nu_i^n$, and hence it is a strong factor in explaining why these points are *similar*. Put in other words, a low $lc_i^j$ value indicates that the axis-aligned bounding-box surrounding $\nu_i^n$ is much narrower along axis $j$ than along the other axes.

To use the above metric to compare dimensions against each other for the same neighborhood and also across neighborhoods, we need to ensure proper normalization. For this, we use compute next the so-called global contributions $gc^j$ of all dimensions $j$, simply by applying Eqn. 6.3 to the centroid of $D^n$ and considering, as neighborhood, the entire $D^n$. Having these values, we define the (normalized) Euclidean ranking contributions of dimensions $j$ at each point $i$ as

$$\mu_i^j = \frac{\overline{lc_i^j}/gc^j}{\sum_{j=1}^{n}\left(\overline{lc_i^j}/gc^j\right)} \in [0,1]. \tag{6.4}$$

Intuitively, the above normalization will give more important (that is, low-valued) contributions to dimensions which are very significant to describe the local similarity of points, but which are less important for describing the overall dataset shape.

**Variance ranking:** An alternative to the above ranking is to use a variance-based solution rather than one using Euclidean distances. For this, we proceed analogously to the previous case. First, we compute the local variance $LV_i^j$ of the $j$ dimensions of all points contained in the neighborhood $\nu_i^n$. Secondly, we normalize these values by the global variance $GV^j = var(\mathbf{p}^j)$ of all values of dimension $j$ over all points $\mathbf{p} \in D^n$. Finally, we compute the variance ranking of dimension $j$ for neighborhood $\nu_i^n$ as

$$\mu_i^j = \frac{LV_i^j/GV^j}{\sum_{j=1}^{n}(LV_i^j/GV^j)} \in [0,1]. \tag{6.5}$$

Analogously to the Euclidean ranking, low values of the variance ranking indicate dimensions which contribute little to the variance of points in a neighborhood, thus dimensions which are good to explain the similarity of these points.

### 6.2.3 Visualizing single top-ranked dimensions

Using either Euclidean or variance ranking, we can produce a set $R_i = \{(j, \mu_i^j)\}$, $1 \leqslant j \leqslant n$ of dimensions and their ranking values for each point $j$. As explained above, we are interested in *low* ranking values, as these show dimensions which can explain best why points in a neighborhood are similar. As such, we next sort $R_i$ increasingly on the values of $\mu_i^j$. The order in which the dimensions $j$ get permuted due to this sorting give us the final *ranks* of the dimensions, from high (important ones, having low $\mu_i^j$ values) to how (unimportant ones, having high $\mu_i^j$ values).

**Color coding:** A first option to get insight in this information is to visualize the top-ranked dimensions $d_i \in \{1, \ldots, n\}$ of each set $R_i$ – in other words, to show, for every single point $\mathbf{p}_i \in D^2$, the most important dimension $d_i$ that explains it similarity with its neighbors. This amounts to visualizing a categorical value (dimension ID) per point. Given the little space available in a (dense) 2D projection displayed as a point cloud, we choose to visualize such values by color coding the projected points by categorical colors indicating the identities of their highest-ranked dimensions.

However, this design would not work when the total number of dimensions $n$ is larger than (roughly) 10, as it is well known that it is hard to generate good categorical colormaps with more entries. To solve this problem, we proceed analogously to the solution outlined in Sec. 5.1.4 for visualizing the best visible dimensions mapped to a projection axis. Specifically, we count, for each dimension $j$, the number of times it is selected as top-ranked over all points in $D^2$. Next, we select the $C = 9$ most-encountered dimensions to color map via a color table produced with ColorBrewer [74]. This essentially highlights the dimensions which are top-rank for *most* points in the visualization. All other top-rank dimensions, which cannot get a color due to the colormap-size limitations, get mapped to a reserved color (dark blue in our case).

**Uncertainty:** If we were to directly visualize the per-point top-ranked dimensions $d_i$ with color coding, as outlined above, the result would show sharp color transitions in areas where the top-ranked dimension quickly changes between neighbor points. Assuming that the coordinates of local neighborhoods in $D^n$ vary (relatively) smoothly, such sharp transitions do not occur, and are misleading. Additionally, in most situations the top-ranked dimension does not explain *all* the similarity of a local neighborhood – in other words, the ranks $mu_i^j$ are rarely all zero except a single one being equal to one.

One way to encode the uncertainty of the top-ranked dimension $d_i$ as being capable of explaining local similarity is to simply consider the values $\mu_i^{d_i}$ of the top-ranked dimensions $d_i$. However, in practice, we have seen that such values are, in general, quite small – this being, possibly, an effect of the so-called curse of dimensionality. To obtain a more robust evaluation of uncertainty, we compute, for a neighborhood $\nu_i^2$, and for every dimension $j$, the sum $\sigma_i^j$ of the rankings $\mu_i^j$ over all points in $\nu_i^2$ for which $j$ has been selected as top-ranking dimension, *i.e.*

$$\sigma_i^j = \sum_{\mathbf{p}_k \in \nu_i^2 \wedge d_k = j} \mu_k^j. \tag{6.6}$$

Finally, we define the certainty, or confidence $\gamma_i$, that the top-rank dimension $d_i$ can explain well a local neighborhood around point $i$ as the normalized value

$$\gamma_i = \frac{\sigma_i^{d_i}}{\sum_{j=1}^{n} \sigma_i^j} \in [0, 1]. \tag{6.7}$$

Simply put, defining the confidence by $\gamma_i$ instead of $\mu_i^{d_i}$ essentially smooths out, or filters, large local variations of $\mu_i^{d_i}$. In other words, neighborhoods having homogeneous top-rank dimensions get a high confidence, while regions having points with different top-rank dimensions get a low confidence. Finally, we note that, in Eqn. 6.6, we use a smaller neighborhood $\nu_i^2$ of radius $\rho_c < \rho$ than the one, of size $\rho$ used to compute the local contributions (Eqn. 6.3). Intuitively put, the larger $\rho$ values allow us to compute the rankings more robustly, while the smaller $\rho_c$ values allow us to capture the fine-grained variation of ranking confidence (see also Fig. 6.2 next).

Having now the top ranks $d_i$ and their confidences $\gamma_i$, we visualize them over the 2D projection using the Voronoi-based (nearest neighbor) interpolation technique presented earlier in Sec. 3.2.2 for the display of projection errors, by encoding top ranks into categorical colors, and using confidences to interpolate between these colors ($\gamma = 1$) and black ($\gamma = 0$). Note that we use nearest-neighbor interpolation instead of the smoother Shepard interpolation (described also in Section 3.2.2) since our data values are categorical dimension IDs, for which interpolation of higher order than nearest-neighbors does not make sense.

Figure 6.2 illustrates the proposed visualization applied to a synthetic dataset of $3000$ points. The points are sampled using a Poisson distribution from three adjacent faces of a 3D axis-aligned cube, and next jittered from their locations by uniformly-distributed random noise of amplitude equal to 5% of the cube size. For projection, we use here PCA [97] rather than more complex techniques, as PCA is simple and easy to interpret, and works well for this manifold-like dataset. Rankings are computed by variance (Eqn. 6.5). The neighborhood size $\rho$ is set to $10\%$ of the diameter of $D^2$. We see that the color-coded projection shows three regions that match very well the three faces of the original cube. In other words, our visual explanation technique has, indeed, succeeded to detect that each such face is best explained by a single dimension. Moreover, we see that points that are close to the cube edges have a low confidence (dark). This is, again, correct, since neighborhoods $\nu_i^n$ centered around such points contain high-dimensional points belonging to *two* faces of the cube, and can as such not be well explained by a single dimension.

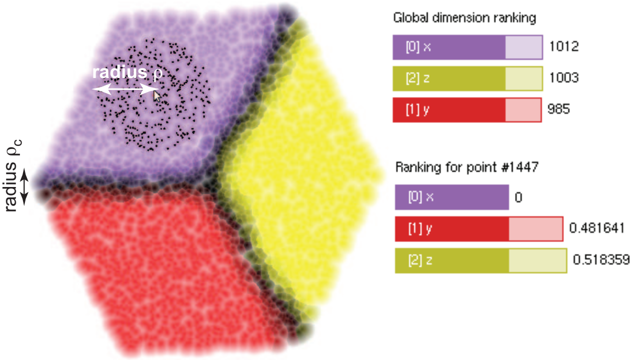


Figure 6.2: Dimension-based explanation of synthetic 3D cube dataset.

To help interpreting the color coding, a legend is added (Fig. 6.2 top-right) to show the specific colors used for the three data dimensions $x$, $y$, and $z$, and also the number of points of the dataset that are (best) explained by each dimension. As visible here, the 3000 input points are split into three approximately equal groups. This is correct, given that the cube's sampling density is relatively uniform, *i.e.*, each of its three faces has about the same number of samples.

To provide more insight into the explanation, we allow brushing the projected points $i$ to show their rankings $\mu_i^j$. This information is shown in the bottom-right legend in Fig. 6.2, for the currently brushed point $i = 1447$, located in the center of the top face. For this point, the legend tells us that $\mu^0 = 0$, *i.e.*, there is no variance of dimension $x$ in the displayed neighborhood around this point. This is correct, given that the respective cube face is orthogonal to dimension 0 ($x$) of the dataset. Additionally, we see that the variances of dimensions 1 ($y$) and 2 ($z$) are relatively equal – which is also correct, given that the respective neighborhood contains roughly the uniform sampling of a plane.

### 6.2.4 Visualizing top-ranked dimension sets

As mentioned above, in many cases it is hard to assign the explanation of similarity of points in a neighborhood to a *single* dimension. So far, we approached this problem by defining a confidence $\gamma_i$ (Eqn. 6.7) to express how dominant the top-ranked dimension is in this explanation. This works well for situations where, locally, a single dimension is responsible for a *large* part of the similarity of points. However, in many cases, a point can have several such dimensions $j$, which differ only very little in terms of values of their (high) rankings $\mu_i^j$. Using the single-dimension explanation will generate a dark uninformative map, since most such points will have, in turn, very low confidence values $\gamma_i$ for their top-ranked dimension $d_i$.

A way to tackle this issue is to extend the possibility of explaining a neighborhood by *several* dimensions rather than a single one. One way to do this is to let the user define the number $k$ of dimensions to be used for explaining each point. However, this solution is impractical, as (1) one may not know what a good value for $k$ is for a given dataset, and, more importantly, (2) different regions in the same dataset may be explained up to the same level by different values of $k$. This phenomenon is analogous to the difficulty of choosing a good representative value for $k$ when computing the $k$-nearest neighbors of a non-uniformly distributed point cloud.

Following the above analogy, a good solution – in the case of nearest neighbors – is to use range search rather than $k$-nearest search. In our context, the analogy works as follows: We define a global maximal ranking value $\tau \in [0, 1]$. Next, for each point $i$, we compute the maximal dimension-set $D_i = \{d_i^j\}_j$ containing all its top-ranked dimensions $d_i^j$, in increasing order of $\mu_i^{d_i^j}$, whose summed rankings $\sum_j \mu_i^{d_i^j}$ is equal or just larger than $\tau$. Intuitively put, all these dimensions $d_i^j$ have a contribution on the inter-point Euclidean distance (or, alternatively, data variance) smaller than a fraction $\tau$ of the total distance.

The above formulation computes dimension-sets $D_i$ of varying sizes for the different points $i$: If, for a point, many of its dimensions have low rankings, then $D_i$ will be large. That is, we need many dimensions to explain why points in a local neighborhood are similar. If, on the contrary, only few dimensions of a point are good explainers (*i.e.*, have low ranking values), then the set $D_i$ will be small. That is, we can explain the similarity of points in a neighborhood by just a few dimensions. Note that the dimension-set explanation converges in the limit to the single-dimension explanation presented earlier in Sec. 6.2.3 – indeed, when either $\tau$ is very small, or only all dimensions (except one) have large rankings, $D_i$ will contain only the first, lowest-ranking-value, dimension. All in all, the dimension-set explanation adapts itself to use as many dimensions as needed to explain a local neighborhood, as specified by the user-given parameter $\tau$. Setting this parameter is intuitive: Small values of $\tau$ imply using, in general, few dimensions for the explanation. This is good for the explanation simplicity, but explains only a relative small fraction of the inter-point distances. Larger $\tau$ values will explain a larger fraction of the inter-point distance (which is good), but also need to use more dimensions (which makes the explanation more complex). In the limit, setting $\tau = 1$ explains the entire distance, but requires using all the $n$ dimensions (as expected). In practice, values $\tau \in [0.02, 0.3]$ have shown to give a good trade-off between explanation accuracy and conciseness.

Once we have computed the dimension-sets $D_i$, we proceed to visualize them similarly to the single dimensions used earlier. That is, given a categorical color table with $C$ entries, we find the $C$ dimension-sets which occur most frequently over the entire projection, and assign to these colors from the table. All other dimension sets (occurring less frequently) are mapped to a reserved color (dark blue).

## 6.3 Examples

We illustrate our dimension-based projection explanations on three datasets (Fig. 6.3). All datasets are projected by LAMP [95], which was selected as it is accurate, simple to use, and was also studied extensively earlier in this thesis. The Euclidean and variance ranking metrics were both tested, and we noticed them to produce very similar results. However, the variance metric was found to be slightly more resistant to noise and parameter variations, so we chose it to use next. The parameters were set to $\rho = 10\%$ of the projection size and $\tau = 0.05$ respectively. Finally, we manually added text labels atop of the same-color regions identified by our explanation, to list the dimensions which were selected for explanation.

### 6.3.1 Wine quality

This dataset has 6497 12-dimensional points. Each point describes a type of *vinho verde* wine [38] by means of 11 physicochemical properties, such as density, concentration of chloride, concentration of sulfur, pH, and alcohol percentage. The $12^{th}$ variable is a (subjective) human-specified level of quality. For more details, we refer to Sec. 5.2.1, where the dataset was introduced.

Projecting this dataset to 2D creates a relatively amorphous shape without clear cluster separation (Fig. 6.3 a). When adding our single-dimension explanation, we see how this shape is partitioned into three zones which are best described by similarities in the *alcohol rate*, *sodium chloride*, and *residual sugar* dimensions (purple, yellow, and red areas in the image). A smaller cluster of brown points is wedged between the yellow and purple zones, being best explained by *volatile acidity*. The borders of the same-color zones appear dark, which is indeed plausible, as points here would be similar with respect to more than a single dimension. Interestingly, these separation borders look relatively smooth, which indicated (indirectly) that the projection succeeds in placing similar points close to each other even at coarse scales.

If we use the dimension-set explanation, the four above regions appear to further split (Fig. 6.3 b). Specifically, *residual sugar* is split into two zones $A_1$ and $A_2$, which are best explained by (*residual sugar*, *free sulfur dioxide*, *total sulfur dioxide*) and (*residual*

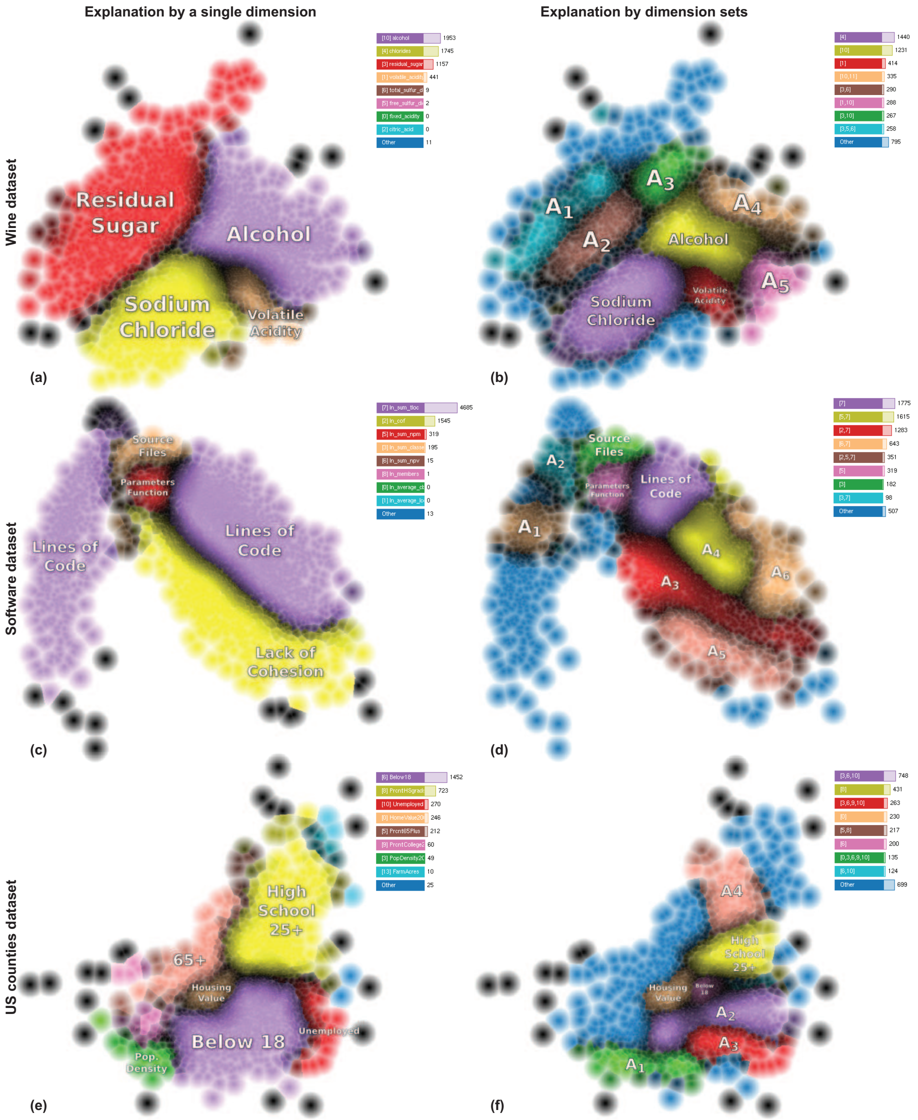


Figure 6.3: Dimension-based explanations of three datasets using a single dimension (left column) and dimension-sets (right column). See Sec. 6.3.

*sugar*, *total sulfur dioxide*) respectively. We see how *residual sugar*, which was previously explaining the large red region in the single-dimension explanation, has been 'inherited' by the dimension-set explanation, which refined it by adding two extra dimensions. Region A$_3$ appears close to the border of the purple and red regions in the single-dimension explanation, and is, indeed, explained by their joint dimensions *residual sugar* and *alcohol*. Similar explanation-refinement phenomena are seen for the regions $A_4$ and $A_5$. The remaining regions, identified by dimensions *sodium chloride*, *alcohol*, and *volatile acidity* in the dimension-set visualization, show points which can be still explained by a *single* dimension, even if we allow multiple dimensions in the explanation.

As compared to the single-dimension explanation, we see considerably more dark-blue points. This is not surprising – when we admit dimension-sets to the explanation, it is very likely that more than $C = 9$ such sets will be created (where $C$ is the size of our colormap). Interestingly, these appear to the periphery of the projection. The reason for this may be related to the tendency of LAMP of generating missing neighbors on the projection periphery (see Chapter 3). As such, 2D neighborhoods around the periphery will correspond to *widely* spread points in the original dataset; so finding concise explanations for such groups of loosely-related points will be hard.

### 6.3.2 Quality of software projects

This dataset describes a corpus of open-source software projects written in C [121]. For each project, 11 software quality metrics were computed, by statically analyzing their source code files, and averaging the results for the entire project. The provided database contains 6733 projects (observations), each having 11 attributes (code quality metrics). A separate $12^{th}$ attribute measures the number of times each project was downloaded. The dataset was introduced in Sec. 5.2.4.

The 2D projection of this dataset exhibits two separate clusters. Using single-dimension explanation (Fig. 6.3 c), the left one is best explained by *total lines of code*, while the right one is split into two groups, best explained by *total lines of code* and *lack of function cohesion*. The double occurrence of the same dimension (*total lines of code*) in two areas of the projection is not erroneous: Brushing the points shows that the respective regions are explained by two different *ranges* of this attribute. Using dimension-set explanation (Fig. 6.3 d) splits the large same-color zones shown earlier into smaller zones, much like in the wine quality example. The left 'lobe' of the projection now becomes mainly blue, which shows that it contains smaller zones that can be explained by a limited number of variables as compared to the right lobe; as such, the explanation algorithm prefers to assign the (few) available colors to the right lobe and leave the left one unexplained. The right lobe is split into several regions: A$_3$ (adds *lack of function cohesion* to the explanation); A$_5$ (adds *lack of function cohesion*, *number of function parameters*); A$_4$ (adds *number of function parameters*); and A$_6$ (adds *number of public variables*).

### 6.3.3 US counties

This 12-dimensional dataset describes social, economic, and environmental data from 3138 USA cities [191]. As for the wine dataset, LAMP projects it into a single clump. Using the single-dimension explanation, we see six large groups, of which the largest two are characterized by the dimensions *below 18* (years old) and *high school 25+* (Fig. 6.3 e). As for the previous two examples, the dimension-set explanation refines these groups into smaller ones (Fig. 6.3 f). The *below 18* group gets split into four parts: A small purple part (still best explained by *below 18*); $A_2$ (adding dimensions *unemployed*, *population density* to the explanation); $A_3$ (adding dimensions *unemployed*, *population density*, and *percent of college/higher graduates*); and $A_1$ (adding dimensions *unemployed*, *population density*, *percent of college/higher graduates*, and *median of owner-occupied housing value*). The yellow region in the single-dimension explanation gets split into a smaller yellow area (still best explained by *high school 25+*); and a pink area $A_4$, which adds *population* $\geqslant$ *65 years old* to the *high school 25+* explanation.

## 6.4 Discussion

Below we discuss several relevant aspects of our explanatory method.

**Related work:** Using color coding to explain salient dimensions in a projection is also used by Gleicher [66] which employs a color field visualization to quickly judge the importance of a dimension in a projected space. However, our way of defining and computing the importance of a dimension, as well as the usage context, are different from the above-mentioned approach.

**Advantages:** Arguably the main added-value of our method is its ability to treat any projection, regardless of the presence of clearly-separated visual clusters herein (see *e.g.* Fig. 6.3a,c). This is in contrast with other clustering-based techniques, which need to separate the projection space into several areas which are next explained. In contrast to this, our method partitions the projection space implicitly. Separately, our method is fully automatic (the parameters $\rho$, $\rho_c$ and $\tau$ working well with the default values mentioned in this chapter). This is in contrast to explicit clustering methods, whose output can depend significantly on clustering parameters, *e.g.*, [175, 130, 101].

**Simplicity:** The proposed technique is easy to learn and use – in most cases, it operates fully automatically, and only requires the user to read the color legends and, optionally, perform some brushing to get more information on the displayed clusters. Computationally speaking, our single-threaded CPU implementation of the method in C++ runs in real-time for datasets of up to 10000 points on a standard PC. As for the error explanatory metrics and 3D explanatory techniques presented earlier in this thesis, our dimension-based explanation is generic, *i.e.*, it can handle any projection technique, as long as we know the high-dimensional coordinates of the projected points.

**Parameters:** If desired, users can control various visual aspects of the explanation using the method's three parameters. Here, *rho* acts as a level-of-detail parameter – large values create larger same-explanation regions, but also typically lower confidence (thicker dark borders), and small values act in the opposite direction. $\rho_c$ acts as a denoising filter: large values eliminate local variations, but create thicker dark borders, and small values work in the opposite direction. Finally, $\tau$ controls the balance between accurate explanations of the inter-point similarity (potentially creating too many subregions which cannot be color mapped) and coarser similarity explanations (which create less subregions, thus minimize the chance that we exhaust the color map entries).

**Hierarchy:** As outlined in the examples in Sec. 6.3, the dimension-set explanation refines the single-dimension explanation in a hierarchical nature. While we cannot state that regions created by the former are *always* a partition of the larger regions created by the latter, spatially speaking, we do see a quite good match here. Also, we see that the finer-grained explanation used by the dimension-set method for some region $A$ always includes the dimension that the single-dimension explanation produced for the region $A'$ that $A$ most overlaps with. In other words, the single-dimension and dimension-set explanations appear to create a two-level hierarchy that partitions the dataset both spatially, and in terms of dimensions used for explanations. Exploiting this observation to generate more refined visual explanation methods is thus an interesting future work direction.

**Limitations:** As our explanatory methods use (categorical) color coding, they are implicitly limited to the maximum size $C$ of such colormaps. As shown in our examples, $C$ can be smaller than the number of regions required for the explanation, both in the single-dimension and in the dimension-set modes. This limitation is expected to become more problematic as the number of dimensions increases. However, in this case, the dimension-set method has the chance of actually working better than the single-dimension explanation, as a single region (using a color entry) can be explained by an arbitrary number of dimensions. Other limitations include the lack of display of values of the dimensions over the identified same-explanation regions. Separately, the Euclidean distance and variance ranking metrics, while simple to compute, may not be the best way one can measure the contributions of dimensions to similarities of close points. In this sense, it is extremely interesting to study feature scoring, dimension correlation, and outlier detection metric well known in machine learning [72, 64]. Fortunately, any such metric can be adapted to be used by our visual explanation, and can be also easily added to the current implementation of our method.

## 6.5 Conclusions

We have presented a simple and automatic technique that visually explains 2D scatterplots, created by multidimensional projections, by the names of the original dimensions. In contrast to other explanatory techniques for multidimensional projections which aim at globally explaining the projection in terms of variances or spreads of attributes, our approach focuses on explaining each local neighborhood of close points in the projection by color-coding the attribute(s) that are most responsible for placing those points close together.

Several directions can be envisaged to extend the local attribute-based explanation of multidimensional projections presented in this chapter. First, automatic segmentation of the regions implicitly created by our ranking techniques can be easily produced using *e.g.* level-set techniques [165]. This would allow explicit construction of labels or glyphs to explain such regions in more detail than by using color-coding only. Secondly, such segments could be used to construct smoother representations of the variation of the explanation confidence atop a single region, by using *e.g.* the construction of generalized shaded cushions following the regions' shapes, along the techniques presented in [182, 26]. Thirdly, our explanations in terms of individual dimensions *vs* dimension-sets creates an implicit hierarchy of coarse-to-fine explanations. Such explanations could be visualized in a single image by adapting hierarchical shaded cushion techniques used for the depiction of treemaps [196, 129]. All these directions open new possibilities for the compact and intuitive explanation of multidimensional projections in ways which are simple to understand for a large class of users. Finally, we intend to better integrate the error-analysis techniques described in Chapters 3 and 4 with the attribute-based explanations, in order to fully support the workflow of first filtering the projection for high-quality regions, then exploring these regions regarding their representative attributes.

## Contributions

The text of this chapter is based on the paper *Attribute-based Visual Explanation of Multidimensional Projections* (R. R. O. da Silva, P. Rauber, R. Martins, R. Minghim, A. Telea), Proc. EuroVis Workshop on Visual Analytics (EuroVA), eds. E. Bertini and J. C. Roberts, Eurographics Association, pp. 134-139, 2015. The key contributions of R. Martins have been the adaption of the dense image-based error visualization technique, presented earlier in this thesis in Chapter 3, to visualize attributes and their confidence levels; the proposal of the variance-based ranking metric (Eqn. 6.5); and the selection, analysis, and interpretation of the software dataset (Sec. 6.3.2).

Chapter 7

# Multidimensional Visual Analysis of Networks

In the previous chapters, we have presented several types of methods that aim to explain, or facilitate the visual interpretation of, multidimensional projections from various angles: Chapters 3 and 4 present methods for explaining projection distance-related and neighborhood-related errors respectively; Chapter 5 presents methods that explain 3D projections in terms of axes related to the original high-dimensional variables; and Chapter 6 explains local neighborhoods in 2D projections in terms of the most important attributes that are responsible for creating these neighborhoods[1].

All the explanatory methods outlined above address projections of multidmensional datasets whose attributes are, essentially, quantitative values. However, as outlined in Sec. 2.1, such attributes can be of more types: integral, ordinal, categorical, text, and relational. From the perspective of our explanatory techniques, integral, ordinal, categorical, and text attributes can be essentially treated in similar ways to quantitative attributes when generating multidimensional projections, by designing suitable dissimilarity or distance functions for the respective attribute spaces, as shown by several papers [1, 24, 23].

In contrast to the above, relational attributes occupy a particular position in the 'attribute space' defining multidimensional datasets. As outlined in Sec. 2.1, their main difference with respect to the other above-mentioned attribute types, is that they are defined on *sets* of observations rather than on individual observations – indeed, this is the essence of the term 'relation'. As such, visually depicting relational attributes creates different, and particularly hard, challenges.

Relational datasets, also called networks or graphs, are ubiquitous in many application areas. There is, for example, a great amount of information regarding human relationships in social networking sites and databases that can be used for various purposes, such as to investigate preference patterns to support commerce and production sectors, to detect and investigate illegal activities, and to discover new forms of communication among individuals [37, 102]. In a different field, the structure, dependencies, and operation of large software systems can be described and analysed as multivariate networks of relationships between entities such as files, packages, components, classes, methods, tasks, developers, change reports, and bugs [46, 45]. Many other application domains generate networks, such as the transportation sector [87], biology [16], and medicine [56].

The most common way to represent networks in visual analysis tools is by using graphs drawn using the well-known node-link metaphor [75, 83], an approach that

[1] The content of this chapter was expanded from the paper *Multidimensional projections for visual analysis of social networks* (R. M. Martins, G. F. Andery, H. Heberle, F. V. Paulovich, A. de Andrade Lopes, H. Pedrini, and R. Minghim), *Journal of Computer Science and Technology*, 27(4):791–810, 2012.

highlights relationships (edges) between actors (vertices) and groups of actors. Various algorithms and tools for graph drawing exist [44, 70, 7]. Recent techniques, such as multilevel layout algorithms [59] and edge bundling techniques [82, 61, 85] succeed in creating drawings of graphs having millions of nodes or edges with limited clutter and good overview of the high-level graph structure.

However, networks may include also *attributes* for their vertices or edges, which can be of all the above-mentioned types (quantitative, intgral, ordinal, categorical, and text). Such attributes are (at least) as important as the relations that describe the network itself to understanding and using the network data. As such, the grand challenge of understanding relational attributed datasets regards the visual depiction of both relations and node-and-edge attributes in a scalable and understandable way.

As we have seen in the previous chapters, multidimensional projections are effective and efficient techniques for depicting large multidimensional datasets, especially when the analysis tasks at hand regard the detection and understanding of groups of similar observations and the detection and analysis of correlations of attributes. As such, it seems natural to consider adapting such techniques for the (more challenging) task of depicting large multivariate networks, *i.e.*, networks whose nodes and/or edges have several data atrributes, and whose analysis should jointly consider the network structure and attribute values. In this chapter, we address this challenge, at two different levels, as follows.

First, we show how the relational data present in a network can be regarded in similar ways to the distance information implied by multidimensional datasets. We next use this approach to construct network layouts, or graph drawings, that emphasizes different kinds of connections between nodes. This approach has similar goals to the well-known force-based techniques used in graph drawing to generate embeddings of graphs [44, 48]. However, as we shall see, several differences exist between our approach and classical force-based graph drawing.

Secondly, we show how to incorporate node attributes in constructing such layouts, so that the resulting graph drawing emphasizes both connections between nodes and node similarities due to their attribute values. When a network is described not only by its relations, but also by attributes mapped on its elements, then it is important that its visual layout reflects not only the connections of the nodes in the network structure, but also the similarities of the nodes' attributes. Creating a graph drawing that reflects strictly the relational information is the common use-case of force-based layouts. Creating a visualization that reflects similarities of observations is the typical use-case of dimensionality reduction methods, or projections. The additional requirement of considering both relations and attributes at the same time, and producing a visual layout that reflects both types of information, is explored in this chapter.

More than one view of the same graph can be generated at once to provide views of different contexts on the same data, and the resulting views can be coordinated to cross-analyse attribute against structure in the network, or even to associate properties of different networks with shared individuals. The goal of the concurrent use of these two approaches is to support data centered visual exploration of networks and to promote easy visual location, in the layout, of groups of nodes that are also related by their properties.

The structure of this chapter is as follows. In Section 7.1, we explore the similarities and differences of relational (graph) data and multidimensional data, with the aim of establishing a conceptual framework that will further let us (a) model relational data as multidimensional data and (b) unify both data types to handle them by multidimensional projections. Section 7.2 presents our use of projection techniques to construct visualizations for multivariate network data. Section 7.3 presents several applications of our proposed visualization methods for the visual exploration of real-world multivariate networks. In Section 7.4 we discuss some of the more important aspects of our proposal to use multidimensional projections for the visualization of attributed networks, and Section 7.5 concludes this chapter.

## 7.1 Preliminaries

Let $G = (V, E = V \times V)$ be a graph, where $V = \{\mathbf{x}_i\}$ denotes its nodes or vertices, and $E = \{(\mathbf{x}_i \in V, \mathbf{x}_j \in V)\}$ denotes its edges respectively. A multivariate attributed graph further associates several attributes to its nodes and/or edges. In this sense, the node-set $V$ is basically identical to our by now well-known set of observations which are the target of multidimensional data visualization methods.

Let us now consider the main high-level goals, or tasks, of graph visualization: For a non-attributed graph, a good drawing thereof should convey (1) the strongly-connected components, or node groups, in the graph; (2) the main connectivity patterns linking the above-mentioned node groups; (3) the paths between (ideally) any pair of nodes in the graph [109, 202]. With due simplification, these tasks can be described in terms of a *similarity* metric: Nodes which are connected should be placed close to each other in the resulting graph drawing; and, consequently, groups of strongly connected nodes should be compact, and placed at a distance from each other that reflects the aggregated connection strength between their node elements. In other words, the connection pattern between nodes in a graph can be modeled as a similarity, or distance, matrix $A = (a_{ij})$ where $a_{ij}$ models the connection strength between nodes $\mathbf{x}_i$ and $\mathbf{x}_j$. In the simplest case, $a_{ij}$ equals one if nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected by an edge, or else it equals zero. In this case, $A$ is also called the adjacency matrix describing the graph $G$. Alternatively, $a_{ij}$ can be set to model the (positive real-valued) connection strength, or weight, of the edge $(\mathbf{x}_i, \mathbf{x}_i)$. These aspects are reflected by many existing graph drawing methods. For instance, force-based methods explicitly model node connections in terms of a distance, or energy, function described by the combination of an elastic spring and repulsion factor [48]. More recent scalable graph drawing methods model the entire graph structure as a distance matrix which is next subjected to a minimization process [103, 104, 62].

In parallel, let us recall the main goal of multidimensional projection methods – that of generating a placement of the observations $\mathbf{x}_i$ in a low dimensional space (typically, 2D or 3D) so that inter-observation distances in this space match the (typically Euclidean) distances between observations in their original high-dimensional space, as given by their attributes. As outlined in Sec. 2.3.1, multidimensional projections can use either the original observations $\mathbf{x}_i$ or alternatively the distance matrix $A = (\|\mathbf{x}_i - \mathbf{x}_j\|)_{ij}$ to compute the desired low-dimensional embedding.

From the above, it follows that, at a high level, the goals of graph drawing and multidimensional projections are related: Both types of methods take as input a matrix describing similarities between observations, and output a low-dimensional drawing where distances between observations should reflect the above-mentioned similarities. As such, it follows that both tasks of standard (non-attributed) graph drawing and multivariate graph drawing could be joined in a single framework based on the usage of multidimensional projections. This observation forms the core of our visualization methods for (multivariate) graphs which are described next.

## 7.2 Method

In line with the preliminaries outlined in Sec. 7.1, we present next two types of methods for the visual depiction of networks. The first type of methods creates graph drawings based solely on the connectivity (relational) information present in a graph, by embedding nodes in the low-dimensional (2D) space so that their distances reflect their graph-theoretic distances (Sec. 7.2.1). The second type of methods adds node attribute information in the process of computing the low-dimensional embedding, and thereby makes the resulting distances between drawn nodes (also) reflect their attribute similarities (Sec. 7.2.2).

### 7.2.1 Connectivity-based projections

Our first proposal for creating a two-dimensional embedding of a network is to map its connectivity pattern to a dissimilarity matrix, designed to reflect the goals of the underlying data analysis. As a running example, we use a bipartite graph $G$ where nodes represent scientific papers and authors of these papers. Edges exist between both types of nodes: Paper-author edges represent authorship of papers; author-author edges model co-authorship of papers; and paper-paper edges represent papers that have common co-authors. We can represent this graph $G$ by an adjacency matrix $A = (a_{ij})$, where $a_{ij}$ has a binary value (0 or 1) indicating whether two nodes (authors and/or papers) are related or not in the senses mentioned above. A more refined approach is to actually encode in the values $a_{ij}$ the strength of connections between nodes. For this, we set $a_{ij}$ to (1) the number of papers two authors have in common, if $\mathbf{x}_i$ and $\mathbf{x}_j$ are both authors; (2) the number of authors two papers have in common, if $\mathbf{x}_i$ and $\mathbf{x}_j$ are both papers; and (3) one, if $\mathbf{x}_i$ and $\mathbf{x}_j$ are a paper and an author thereof, respectively.

With the adjacency matrix $A = (a_{ij})$ described above, we next construct two typesof dissimilarity matrices to create a two-dimensional embedding of the underlying research network using projection techniques, as follows:

1. **Modified adjacency matrix:** We construct a modified adjacency matrix $A' = (a'_{ij})$ where $a'_{ij} = 1/a_{ij}$ if $a_{ij} \neq 0$ and otherwise $a'_{ij} = k \cdot \max_{i,j} a_{ij}$, where $k$ is a fixed constant value. The matrix $A'$ is essentially a distance matrix between papers

and/or authors – if two papers and/or authors have a strong connectivity, in the sense implies by the original adjacency matrix $A$, then they will have a small distance in $A'$. We next use this distance matrix as input for any projection method that accepts distance matrices, such as the multidimensional scaling (MDS) methods overviewed in Sec. 2.3.1. As outcome, the resulting projection will place nodes which are *immediately* adjacent close to each other. In contrast, nodes which are connected by paths having a length superior to one are not guaranteed to be placed at distances proportional to the connecting path-length.

2. **Shortest-path matrix:** Consider the same graph $G$ as outlined above. For this graph, we construct a distance matrix $A' = (a'_{ij})$ where $a'_{ij}$ equals the length of the shortest path connecting nodes $\mathbf{x}_i$ and $\mathbf{x}_j$, if such a path exists. If the respective nodes are not connected by any path in $G$, then we set $a'_{ij} = k \cdot \max_{i,j} P$, where $P$ is the maximal path length between any two nodes in $G$. By using this distance matrix $A'$ as input for a multidimensional scaling method, the resulting projection will reflect more the global path-length structure of $G$ rather than small-scale connectivity patterns, such as captured by the proposal listed at point (1) above.

The examples in Fig. 7.1 illustrates the characteristics of the connectivity-based and shortest-path-based projections using IDMAP [122]) as a projection technique. The input network is a dataset called *VisBrazil* which describes the publications in the field of data visualization in Brazil for a period of 7 years (2003–2010). In detail, this heterogeneous network includes 262 nodes (105 papers and their 157 authors). The network has 1861 edges, which represent author-paper relations (representing the authorship of the papers), author-author relations (representing co-authorship of papers), and paper-paper (representing co-authors of papers). For this dataset, a paper can be thought of as a community of authors. The final drawings of the projected observations encode the observation type into shape – authors are circles and papers are square glyphs, respectively. The size of the nodes maps their *betweenness centrality*, a measure that indicates how many shortest paths between all pairs of nodes in the network pass through a given node, or, in other words, the importance of the node in the exchange of information between the nodes of the network [57, 19]. In this case, since the nodes are authors and their co-authorships in published papers, nodes with high betweenness centrality indicate important collaborators between different research groups (the exact centrality value reflected by the nodes' sizes is not important, only the comparison between small and high values). In both the modified adjacency matrix plot (Fig. 7.1a) and the plot using the shortest-path matrix (Fig. 7.1b), we can see that highly central nodes (in terms of their centrality metric) are also located quite close to the local center of the projection. Additionally, author nodes (circles) are color mapped to show their productivity in the network, *i.e.*, the number of published papers, using an inverted heat (yellow to dark-brown) colormap. For papers, we set the color to correspond to a value of zero (light yellow). The fact that most of the large (author) nodes are also darker indicates that there is a direct relationship between collaboration and productivity in this network.

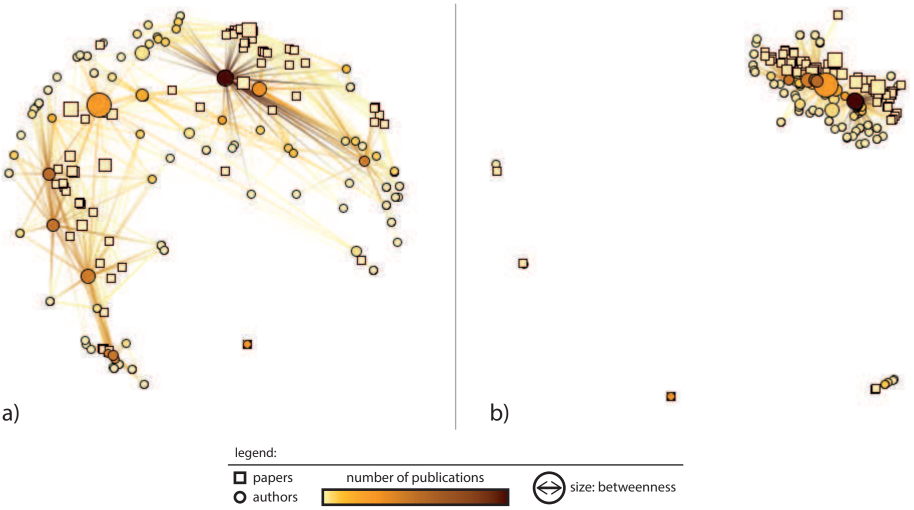


Figure 7.1: Connectivity-based projections of the *VisBrazil* network constructed using the IDMAP projection technique [122]. (a) Layout based on the modified adjacency matrix. (b) Layout based on the shortest-path matrix.

From a first look at Figs. 7.1a-b, one feature becomes apparent – the presence of isolated points that populate empty areas around the large interconnected groups of points. Browsing these points, to show their data attributes, we find out their reason of showing up as outliers: These are small connected groups of papers and/or authors that do not share any co-authorship with the remaining large group of papers and authors. This feature can be found in most graph-based bibliography analyses: There is usually a large connected component, explained by the propagation of cooperation between researchers; the isolated graph fragments represent small sets of papers of research groups with separate interest areas or papers of newly formed research groups. As can be seen in Figs. 7.1a-b, both types of distance matrices proposed above isolate such smaller groups from the larger main component quite well.

The projection method we are using (IDMAP) is specifically developed to consider distances between all pairs of points present in its input dataset, in line with most other distance-matrix-based projection methods. Also, we note that there is a large difference in connectivities (in the sense of number of edges, or node degrees) between nodes present in the large central strongly-connected component and the isolated small components. As such, nodes in these small groups will have large distances to all other nodes, given by $k \cdot \max_{i,j}(a_{ij})$, as described earlier. Consequently, these small groups will be placed far away from the central strongly-connected component, leaving large areas of (unused) white space to surround them.

We continue the analysis of the connectivity-based projections by focusing on the main inter-connected group of authors from the *VisBrazil* network, comprising 222 nodes (89 papers, 133 authors) and 1739 edges (see Tab. 7.1). We call this selected subset of observations *VisBrazil-Main*. The visualization of *VisBrazil-Main*, created with the same parameters as the visualization of *VisBrazil* (Fig. 7.1), is shown in Fig. 7.2.

Table 7.1: Differences between the datasets *VisBrazil* and *VisBrazil-Main*

| **Datasets** | **Nodes** | | | **Edges** |
|---|---|---|---|---|
| | Papers | Authors | Total | |
| *VisBrazil* | 105 | 157 | 262 | 1861 |
| *VisBrazil-Main* | 89 | 133 | 222 | 1739 |
| **Difference** | 15,23% | 15,28% | 15,26% | 6,55% |

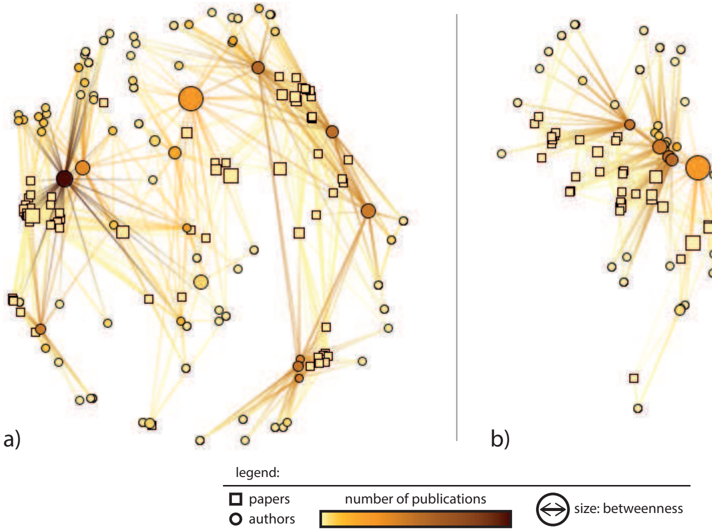


Figure 7.2: Connectivity-based projections of the *VisBrazil-Main* network, formed by the main interconnected group of authors from *VisBrazil*, using IDMAP [122]. (a) Layout based on the modified adjacency matrix. (b) Layout based on the shortest path matrix.

In Fig. 7.2, we see that the relationship between the nodes' computed centralities, encoded by their sizes, and their spatial positions in the central areas of the layout, are more apparent for the shortest-path-matrix case (Fig. 7.2b) than in the modified-adjacency-matrix case (Fig. 7.2a). This insight is useful to help us understand how the shortest-path-matrix plot actually works: Nodes with high centrality – those with short paths to most other nodes in the graph – are pushed towards the center of the projection, which leads to the interpretation that central nodes are hubs that connect peripheral nodes. Separately, let us consider the position of the nodes which have a high centrality *and* high number of published papers, *i.e.*, the large-and-dark circles. We notice that their organization in the two layouts (Figs. 7.2a-b) changes significantly: In the shortest-path-matrix plot, these nodes are located, again, close to the local center of their surrounding node distributions, confirming their positions as central interaction hubs int his social network; in contrast, in the modified-adjacency-matrix plot, they are located at various positions with respect to the surrounding nodes, thereby making it harder to visually detect them as important interaction hubs. Following these insights, we argue that projections created by shortest-path distance matrices are more effective in highlighting the distribution of nodes, in such social networks, around central communication hubs.

**Combining projections and spring embedders:** Besides being able, on their own, to construct two-dimensional embeddings of networks, our projection-based techniques are also useful as 'preconditioners' for classical force-based layouts used to visualize such networks. We illustrate this in Figure'7.3. The input dataset consists of our entire *VisBrazil* network discussed above. Figure 7.3a shows a graph layout created by using a standard spring embedder that iteratively adapts the positions of the nodes, starting from random placements, until the total spring energy captured by the graph's edges, falls under a given value, or until a maximal number of 30 iterations has been performed [48]. Figure 7.3b shows a layout of the same graph, using the same spring embedder and parameters, this time starting from node positions given by the modified-connectivity-matrix method (Fig. 7.1a). As can be seen in Fig. 7.3b, when a force-directed layout is applied after a projection of the nodes, there is less likelihood of large groups of connected nodes 'collapsing' together in the layout, as it happens with a conventional force-directed layout with random initial node placement (Fig. 7.3a). We shall analyse this property in more quantitative terms later on in Section 7.3. Projecting nodes first before performing the force-directed layout prevents local energy minima, in the force-directed layout, from derailing force-based placements. Importantly, we note that even a force-based placement where spring stiffnesses of the edges are given by connectivity calculation can not produce such a separation effect. As such, our proposed use of a projection step as preconditioner for applying a force-directed graph layout is of added value.

The movements of the nodes in Fig. 7.3b by the force-directed layout starting from node positions determined by our projection reveal a better picture of small sub-graphs to the left and right of the larger subgraph. These small subgraphs represent small research groups working in relative isolation. Such subgraphs are also visible in the original projection (Fig. 7.1a) in the sense of being separated from the central strongly-connected graph component, due to the large distance values we set in the modified adjacency matrix $A'$ between disconnected nodes. However, by using the projection only, these small subgraphs appear as 'collapsed' to sets of overlapped nodes. In contrast, the force-directed pass used after the projection keeps these components separated from each other, but also spreads their nodes more uniformly over the available space, therefore decreasing clutter and overlap (Fig. 7.3b). Finally, we note that, by using a force-directed layout only, starting from random initial positions, nodes in the above-mentioned small disconnected components get mixed with other nodes (Fig. 7.3a). To separate these, a large number of force-directed iterations is required – and even if such an iteration count is allowed, separation is not guaranteed, which in turn leads to a cluttered layout. At a high level, the explanation of the above phenomena is as follows: Both our IDMAP projection and force-directed technique are, essentially, *embedders* that take as input

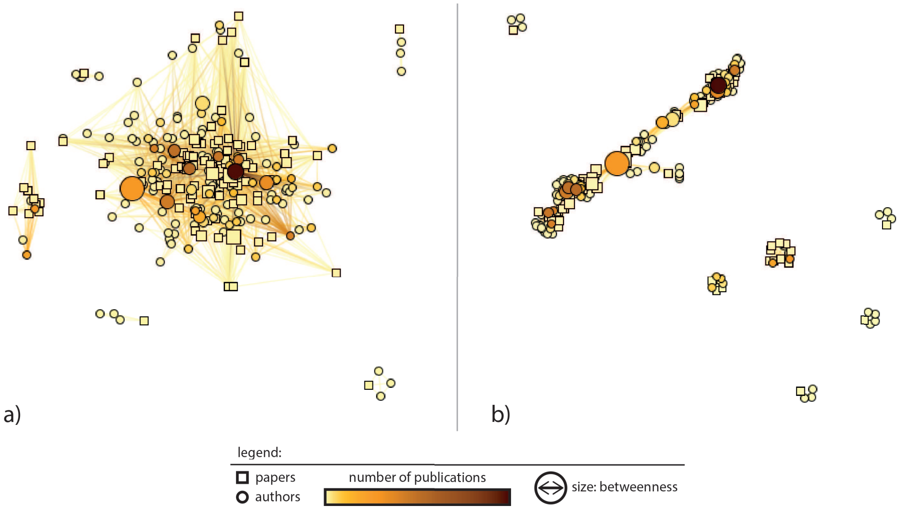


Figure 7.3: Using connectivity-based projections of the *VisBrazil* network as a preconditioner for a force-directed layout. (a) Force-directed layout starting from random node positions. (b) Force-directed layout starting from node positions given by a projection based on our modified adjacency matrix (Fig. 7.1a).

a distance matrix and aim to create a two-dimensional configuration of nodes whose 2D distances best approximates the given distance matrix. However, differences exist between the working and heuristics of these two techniques. In a general sense, neither of them is ideal: IDMAP may preserve well distances from the input matrix, but this may cause nodes to be placed too close to each other when such distances are too small; the spring embedder, on the other hand, achieves better clutter and overlap minimization by its repulsive forces [48], but cannot preserve distances between nodes as well as IDMAP. In this sense, the combination of IDMAP (or, in the more general sense, a multi-dimensional projection) and a spring embedder achieves a good combination between desirable features of both types of techniques.

**Comparing projections and spring embedders:** Our final example, shown in Fig. 7.4, shows a layout of the *VisBrazil-Main* dataset, *i.e.*, the central strongly-connected component of our authorship network, generated by 100 steps of a force-based spring embedder starting from random initial positions. Interestingly, the result shown in Fig. 7.4 reflects a spatial distribution of the nodes which is quite similar to the one shown by the projection based on the shortest-path-matrix (Fig. 7.2b): Both images show that the network is roughly divided into two large groups, to the left and to the right of the projection, with a third smaller group towards the bottom, mediated by a single well-connected researcher (marked as Researcher #1). However, the force-based layout (Fig. 7.4) consistently settles on smaller edges between the nodes, cluttering the organization inside the two distinct groups and making it hard to distinguish between papers (squares) and authors (circles). In contrast, the projection based on the shortest-path-matrix (Fig. 7.2b) shows

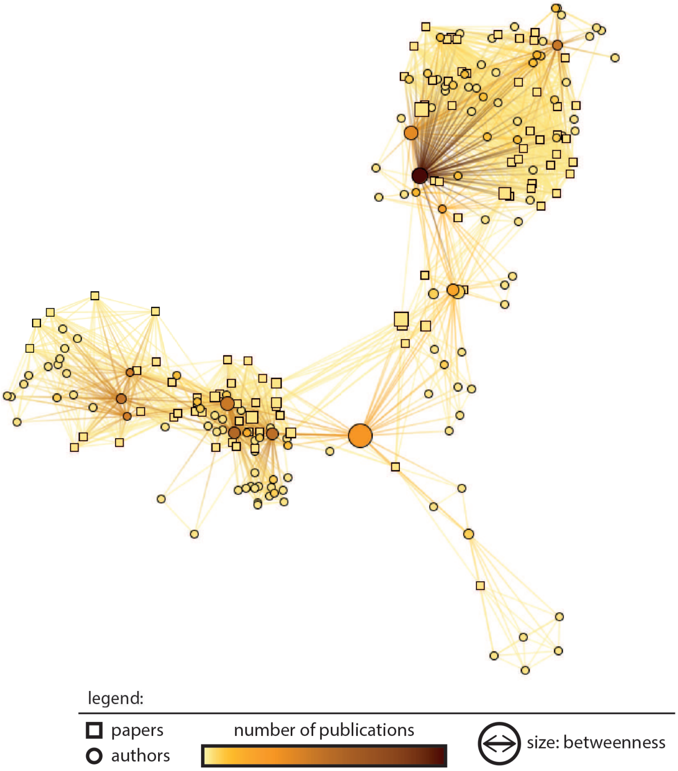


Figure 7.4: Connectivity-based projection of the *VisBrazil* network – 100 steps of force-based layout with random initial positions.

a clearer separation between nodes of different types, which makes it easier to spot their effect on the network around them. Together with the results presented in Fig. 7.3, this supports our claim that projection-based techniques are good alternatives and/or preconditioners to force-based embedders for the creation of low-clutter graph layouts.

### 7.2.2 Attribute-based projections

When a dataset is composed not only of relations, but also of (potentially multiple) attributes that describe each observation (node) and/or relation (edge), a complete visual analysis of this dataset must involve not only the exploration of its structural connectivity patterns, but also of the similarities between its elements in terms of their attributes. For this, the previously described techniques (Sec. 7.2.1) are not enough, as these only consider the relations. To accommodate attributes, we consider separately the two cases of edge attributes and node attributes. Edge attributes are relatively easy to accommodate using the distance-matrix based techniques outlined in Sec. 7.2.1, by weighting the entries $a'_{ij}$ of the connectivity-based distance matrix $A'$ by the values of such attributes. This follows the typical usage of edge attributes to model the importance, or strength, of an edge, which is in line with the semantics of the connectivity-based distance matrix entries. When multiple attributes are defined on an edge, they can be aggregated by *e.g.* summation or averaging to compute a single weight. Node attributes, however, are slightly less straightforward to integrate in the above solution, since they are defined per observation rather than per relation. As such, we focus the discussion in this section on node attributes solely.

We propose in this section an approach to combine, in one or more visualizations, the two different viewpoints of a given network given by its relations and node attributes. For this, we shall merge a number of distance matrices that capture separate aspects of the network into a single distance matrix that reflects the user's analysis goals. This matrix is next used as input for a MDS-type projection technique, in the same way as already described in Sec. 7.2.1.

To demonstrate the underlying idea, we first describe the dataset to be visualized. This dataset, called next *VisBrazil-Papers*, is a subset of the *VisBrazil* dataset introduced in Sec. 7.2.1, containing only the paper-type nodes (that is, no author nodes) and edges linking them, which represent, as already explained, papers that share at least one common co-author. Edges are weighted by the number of common co-authors of their paper nodes. To this relational information, we can add extra data concerning the *contents* of the papers. For this, we transform their text contents into multidimensional attribute vectors by using the classical Vector Space Model (VSM) representation of text documents [154]. In short, this process extracts all words from each document, removes stopwords, transforms the remaining words into so-called radicals, and removes radicals that occur less frequently over the entire set of processed documents. Each document then gets an attribute vector with one entry (attribute) per radical having, as value, the number of times this radical occurs in the respective document (a similar approach was used for the *ALL* dataset used in Sec. 5.1). Following this description, it becomes clear why we eliminated authors from the dataset – these do not have the same types of attributes as papers.

**Connectivity *vs* attribute projections:** Our first question is how does a projection that solely uses the relational information in this dataset differ from a projection that solely uses node-attributes. Figure 7.5 shows two visualizations that illustrate these differences. Both visualizations are created by using the IDMAP projection. Figure 7.5a uses a distance matrix computed from the VSM node-attribute representation, where attribute-vectors are compared by using cosine distance, as typical in document processing. Hence, distances between the nodes in this image encode the dissimilarity of the documents' texts. Figure 7.5b uses the shortest-path distance matrix encoding the relations in the network (Sec. 7.2.1). Hence, distances between nodes in this image encode the connectivity patterns in the network.

To make the comparison of the two projections in Fig. 7.5 easier, we subsequently cluster the nodes based on similarity of their VSM attributes, using the Bisecting K-Means algorithm [174] (other clustering algorithms can be used equally well), and next color all nodes in each cluster by the same categorical color. Since the clustering is identical for both projections, differences in the distribution of node colors in the two images indicate differences in the way the two projections place nodes in the two-dimensional

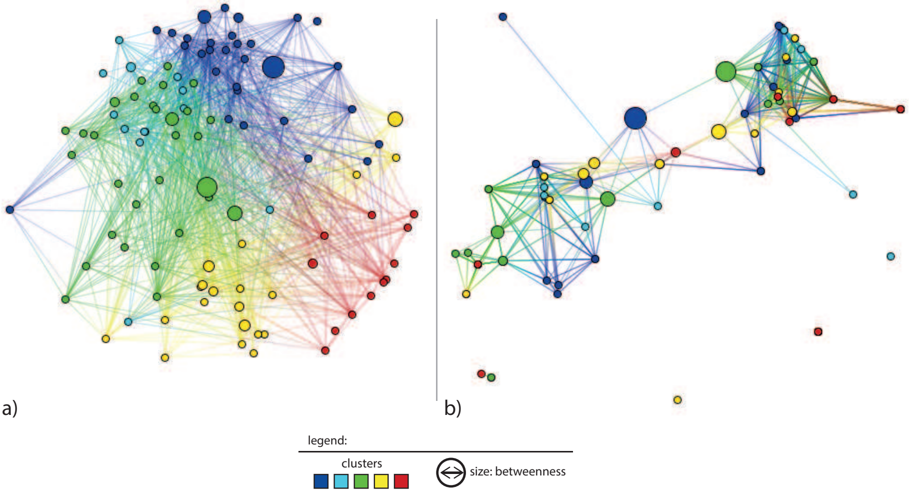


Figure 7.5: Two IDMAP projections of the *VisBrazil-Papers* dataset. (a) Projection using only the VSM node attributes. (b) Projection using only the shortest-path distance matrix encoding connectivity.

embedding space. We notice that same-color points, which belong to the same cluster, in Fig. 7.5a are placed near each other, reflecting the fact that the layout favors similarities in the multidimensional attributes. In contrast, there is no visible relation between node colors and positions in Fig. 7.5b. Indeed, this image is a projection that places connected nodes close to each other, regardless of their attribute values. On the other hand, the layout in Fig. 7.5b has shorter edges and less edge-crossing, since, as explained, this projection follows the network's connectivity patterns. In contrast, edges in Fig. 7.5a are long and crossing all over the projection, since this projection completely ignores the relational data in the input dataset. This comparison shows two extreme viewpoints that can be obtained from the same dataset, by considering its node attributes or its node connections. Both viewpoints have their merits: Figure 7.5a is preferred if one is interested purely in an attribute-based analysis; while Figure 7.5b is preferred if one is interested purely in a connection-based analysis.

**Combining connectivity and attributes:** There are, clearly, cases when one wants to analyse this type of dataset from a perspective that regards both attributes and connections. For this, we need to merge both sources of information in the construction of our projection. A simple way to do this is to compute two distance matrices, one using only the connections (as explained in Sec. 7.2.1) and the second one using the cosine distance between all attribute-vectors of all nodes; next, we add these distance matrices to yield a final distance matrix that we use to compute the projection. To stress connectivity *vs* attribute similarity, we can next weigh the two distance matrices in the summation.

Figure 7.6 shows, for the *VisBrazil-Papers* dataset, the results of combining the connectivity and attribute distance matrices, with equal weights, and applying the

IDMAP projection technique to the final matrix. Colors follow the same scheme as in Fig. 7.5. Figure 7.6a shows the result when connectivity is captured by the shortest-path distance matrix. Figure 7.6b shows the result when connectivity is captured by the modified adjacency distance matrix. Several observations can be made here. First, both images in Fig. 7.6 have a structure that appears to be an in-between form of the extreme structures shown in Figs. 7.5a,b: Same-color nodes are located relatively close to each other (like in Fig. 7.5a, when using only node attributes in the projection); and strongly-connected node groups are placed close to each other (like in Fig. 7.5b). Additionally, when using the shortest-path distance matrix to encode the network's connectivity (Fig. 7.6), we see how central nodes (large circles) are placed far from the layout's periphery.

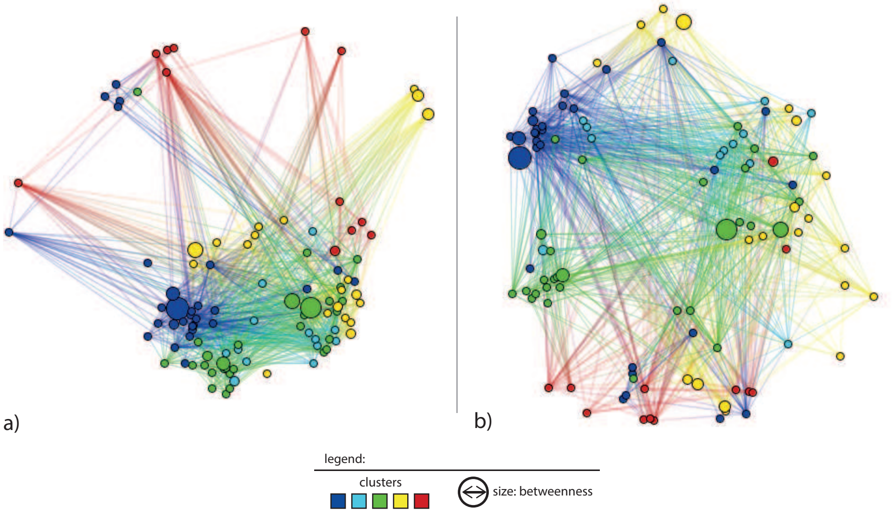


Figure 7.6: IDMAP projection of the *VisBrazil-Papers* dataset, based on a combination of the nodes' attribute distance matrix and connectivity modeled by the (a) shortest-path distance matrix and (b) modified adjacency matrix.

It might be the case that, depending on the analysis goals, one of the two factors – connectivity or attributes – is more important than the other. To reflect that in the resulting visualization, we can change the weights of the corresponding distance matrices when summing them up to compute the final distance matrix. Figure 7.7 shows this by using the same dataset and settings as in Fig. 7.6a, but having different weights on the two distance matrices. In the first image (Fig. 7.7a), we set the weight of the attribute distance matrix to be 3 times larger than the weight of the shortest-path matrix that encodes connectivity. As a result, same-color (thus, similar) nodes are placed closer to each other than in Fig. 7.6a, the result approaching more the purely attribute-based image in Fig. 7.5a. In the second image Fig. 7.7b), we set the weight of the shortest-path distance matrix to be 3 times larger than the weight of the attribute distance matrix. As a result, strongly-connected node groups are placed closer to each other, and the placement is less influenced by the node attribute similarities. This can be seen by observing the spread of the green nodes in the projection, which, even though similar, are pulled to relatively far apart positions by their local connections to other nodes.

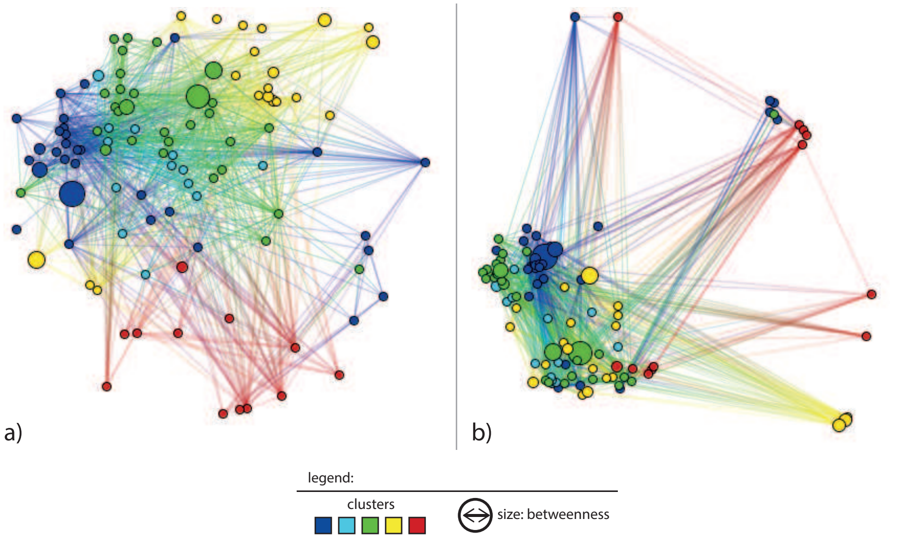


Figure 7.7: IDMAP projection of the *VisBrazil-Papers* dataset, using as input a weighted combination of attribute and shortest-path distance matrices. (a) Attribute distance matrix has a weight of 3; (b) Shortest-path distance matrix has a weight of 3.

Merging of the attribute and connectivity distance matrices does not have to include necessarily *all* the node attributes of the input dataset. Depending on the analysis goals, when exploring a network the user might be interested in the similarity of only one (or a few) attributes, instead of all of them. As such, only these attributes should be included in the computation of the projection. This can be easily done by considering only these attributes when computing the attribute distance matrix, prior to merging it with the connectivity distance matrix. Figure 7.8 shows such an example. The dataset and projection technique here are *VisBrazil-Papers* and IDMAP, respectively, like in our earlier examples. The projection uses a distance matrix merging the shortest-path distance matrix with an attribute distance matrix considering only the *year* attribute of each node (paper). Nodes are colored by their publication year, ranging from 1998 to 2010, and scaled by their betweenness centrality. The resulting projection shows clearly how nodes are positioned to reflect their connectivity pattern, with high-betweenness nodes closer to the layout center, expected. In the same time, we see that same-color nodes are placed quite close to each other – hence, the projection captures well similarity of the *year* attribute.

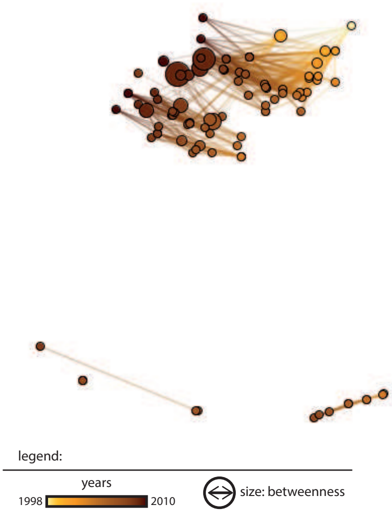


Figure 7.8: IDMAP projection of the *VisBrazil-Papers* dataset, created by a combination of shortest-path distance and year-attribute-similarity matrices. Colors encode values of the publication-year attribute, ranging from 1998 to 2010.

## 7.3 Applications

We next describe several applications of our techniques for visualizing multivariate attributed networks. All applications take the form of using our techniques for exploring a real-world dataset containing both relations and node attributes, and describing relevant insights found by our visual exploration. Sections 7.3.1, 7.3.2, and 7.3.3 address the utilization of network visualizations constructed using purely connectivity information (Sec. 7.2.1). Sections 7.3.4 and 7.3.5 address the utilization of network visualizations constructed using the combination of connectivity information with node attribute values (Sec. 7.2.2).

### 7.3.1 Connectivity-based projections: Research networks

The analysis of networks representing interactions among researchers and their research subjects is an application useful for several purposes, such as understanding the evolution of a subject area and evaluating impact, influence or performance from a certain point of view. This example is meant to illustrate the nature of displays and types of analysis provided by connectivity-based projections.

The dataset, obtained from [3], is the collection of all citations of two major journals in areas of computer graphics: IEEE Computer Graphics and Applications (CG&A) and Computer Graphics Forum (CGF). The full graph of citations and authors extracted from this dataset comprises 2471 papers and 3841 authors. To explore this graph, we computed its modified adjacency matrix, and used this matrix to project the nodes to 2D, using the IDMAP projection technique.

Figure 7.9 shows the first projection-based visualization of this dataset. We see how the modified adjacency matrix distance-technique dedicates most of the available 2D space to the large connected subgraph of authors and papers (the crescent shape in Fig. 7.9a). We also see an isolated outlier, which represents the projection of all papers and authors which are not connected to the large crescent shape. These outlier nodes are actually the ones that determine the crescent-like shape of the projection – in other words, for the highly-connected nodes in the bulk of the graph to be placed (equally) far away from the disconnected outliers, the bulk needs to assume the crescent shape visible in the projection. In Figure 7.9a, nodes are colored by their degree (number of edges), using a blue-to-red colormap. Figure 7.9b shows the same projection, this time drawn without the edges that connect the nodes; also, node colors show now the nodes' betweenness centrality (introduced in Sec. 7.2.1), using a red to blue colormap. We see here how highly central nodes, *i.e.* papers and/or authors that are highly cited, are distributed over the layout of the projection, acting as 'connection hubs' between all nodes in the network.

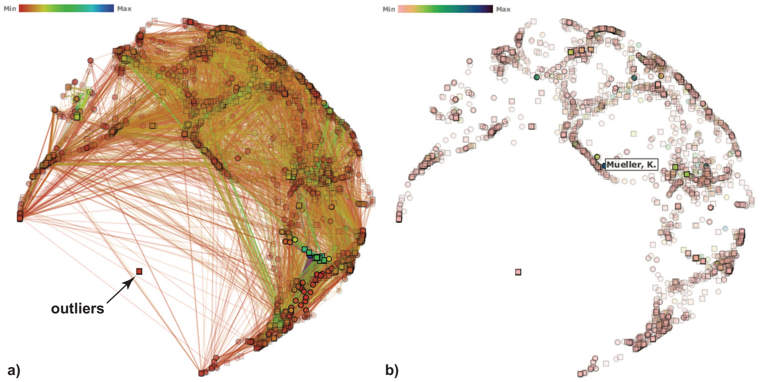


Figure 7.9: Paper-author network for the CG&A and CGF journals, composed of 2471 articles and 3841 authors. (a) Projection colored by graph degree; (b) Projection (drawn without edges) colored by betweenness.

The above images are useful for locating the outlier papers and authors, as being items which are disconnected from the main body of related publications. However, as we have seen, the modified adjacency matrix technique collapses all these outliers to a single location, making it impossible to further analyse potential relations between them. To better see and analyse the structure of the disconnected outlier component, we apply a force-directed layout to the paper-author graph, initialized with the node positions delivered by the projection shown in Fig. 7.9. Figure 7.10 shows the result. In the left image (Fig. 7.10a), we see how the disconnected outliers, which were originally collapsed to a single 2D location, are now spread over a larger area of the available 2D space, while the crescent shape of the bulk of related publications is kept. This allows us to better see the structure of the outlier component – or, in other words, the available projection space is used more efficiently, in the sense that less whitespace is spent in showing that the outlier component is disconnected from the bulk component. This result is not surprising: Projection techniques, such as IDMAP which was used here, try to approximate the input distances, provided in the distance matrix fed to the projection algorithm, in the positions of the 2D points. Given the way our modified adjacency matrix is computed (Sec. 7.2.1), it is not surprising that disconnected components are placed far away from the remaining nodes in the projection, since they are marked as being at very large distances from all nodes not being connected to them. In contrast, force-directed layouts care far less about an *accurate* preservation of high-dimensional distances. Moreover, the node-to-node repulsion terms used in typical force-directed algorithms [48] spread the nodes away from each other, thereby making a more effective use of the available empty space in the embedding dimension.

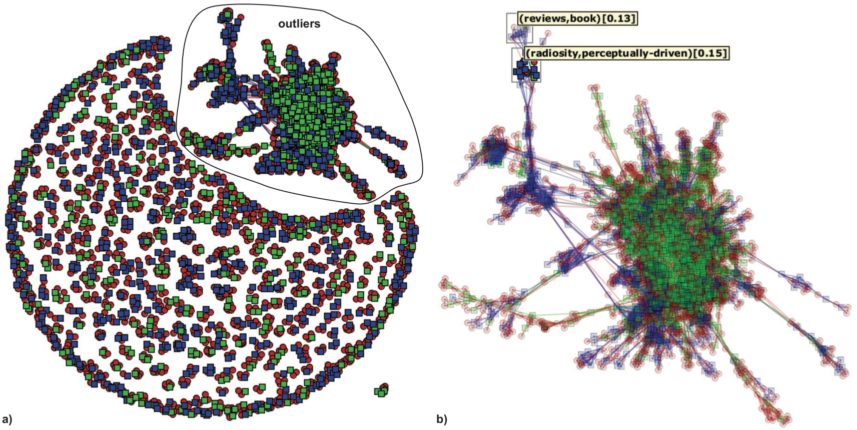


Figure 7.10: Paper-author network for all papers for the CG&A and CGF journals, laid out with a spring embedder initialized by the projection layout in Fig. 7.9. (a) Entire network, colored by journal ID. (b) Zoom-in on the disconnected outlier component in Fig. 7.10a.

### 7.3.2 Connectivity-based projections: Quality analysis

As shown in Sections 7.2.1 and 7.3.1, networks can be visualized by using multidimensional-scaling-type (MDS) projections based on their modified adjacency matrices. The examples given in the above-mentioned two sections show that such visualizations look, globally speaking, similar to visualizations created by classical methods that exploit the available relational information, such as force-directed methods. However, such insights are qualitative at best. To justify the use (and added value) of projections as tools for constructing two-dimensional embeddings of networks, more detailed measurements and comparisons of the quality of the produced layouts needs to be done.

In this section we address the above task of quantitative exploration of the quality of graph layouts created by using our modified adjacency matrix. The proposed procedure comprises the selection of a number of real-world network datasets; the creation of two-dimensional graph layouts for these datasets using (1) standard force-based techniques where node positions are randomly initialized; and (2) force-based techniques where node positions are initialized based on our proposed projection; and the comparison of a number of quantitative metrics, measured on both types of visualizations, in order to compare their quality.

For our analysis, we employed three datasets of co-authorship networks: The *eurovis* dataset includes all papers and authors of the proceedings of the EuroVis Conference on Data Visualization, and its precursor symposium VisSym, from 1990 to 2010. The second dataset (*vis*) includes all papers and authors of the proceedings of the IEEE Visualization and IEEE InfoVis conferences, from 2001 to 2010. The third dataset (*agric*) includes papers (and their corresponding authors) published from 2009 to 2011 by Brazilian researchers on agriculture. Table 7.2 summarizes the sizes of the graphs formed from these datasets. The last column is the percentage of nodes that belong to the larger connected component of each dataset.

Table 7.2: Datasets used for the analysis of quality of connectivity-based projections.

| **Dataset** | **Papers** | **Authors** | **Nodes** | **Edges** | **% Largest Component** |
|---|---|---|---|---|---|
| *eurovis* | 420 | 865 | 1,285 | 4,838 | 59.1 |
| *vis* | 1,586 | 2,849 | 4,435 | 21,176 | 72.2 |
| *agric* | 2,220 | 3,958 | 6,178 | 30,965 | 73.2 |

For each of the above three datasets, we generated layouts based on classical force-directed methods (further called *force*) and force-directed methods initialized by our projection technique (further called *proj-force*). Figure 7.11 shows two such typical layouts, for the *eurovis* dataset. As visible, and in line with the similar results shown in Sec. 7.2.1, the two types of layout methods yield very different results. However, from a visual investigation only, it is not possible to tell which of the two layouts has a better quality. To measure quality in a more detailed and objective way, we measure next three metrics, for each pair of layouts (*force vs proj-force*): two metrics quantifying the distribution of nodes based on their connectivity, and the number of edge crossings in the resulting layout. The metrics are averaged over five runs of each dataset and each algorithm, to account for possible variations. The aim of this experiment is to compare these metrics for the *force* and *proj-force* techniques over the three considered datasets, and thereby quantitatively assess the added-value, in terms of quality improvement, of using our projection technique as initializer for a force-based placement.

Figure 7.12 shows, for our three considered datsets, the first two quality metrics we consider, which are the neighborhood preservation (*npres*) and neighborhood hit (*nhit*), described in detail below.

**Neighborhood preservation (*npres*):** Given a number of $k$ nearest neighbors, *npres* measures the percentage of neighbors, averaged for all nodes, which are the same in the two-dimensional layout as well as in the adjacency matrix $A$ introduced in Sec. 7.2.1. Higher *npres* values are, clearly, desirable. Preserving node neighborhoods in two-dimensional layouts is, clearly, not an easy task in the case of highly connected graphs.

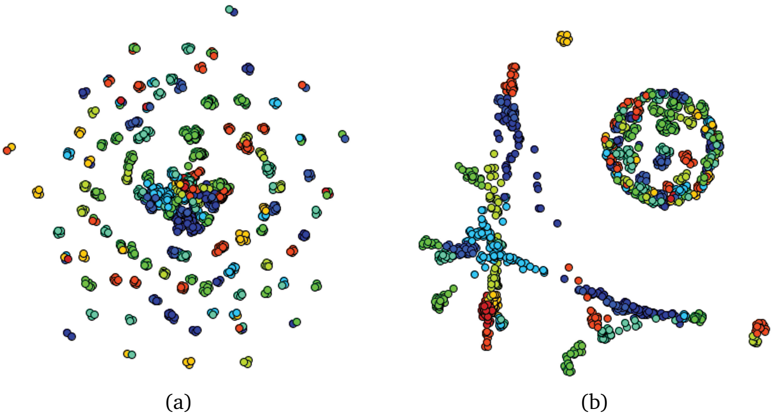


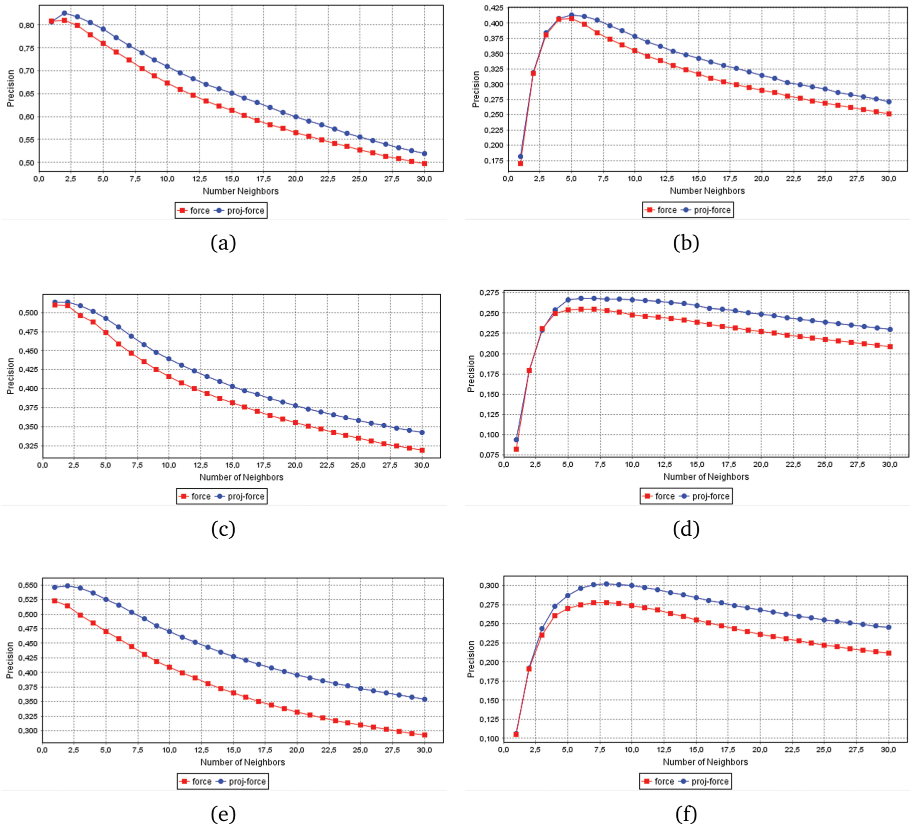
Figure 7.11: Layout for the *eurovis* dataset by (a) force-based method using random positions (*force*); and (b) projection followed by force-based method (*proj-force*). Colors indicate the $k$-means clusters in the dataset.

However, as the *npres* plots in Fig. 7.12(b,d,f) show, node neighborhoods are consistently better preserved when using a force-directed method preconditioned by a projection (*proj-force*) than when using a force-directed method alone (*force*).

**Neighborhood hit (*nhit*):** We first cluster the points using a distance-based clustering method – $k$-means in this case. Then, we count, for each node $\mathbf{x}_i$, the number of its 2D $k$-nearest neighbors, with $k$ being a free parameter, that fall into the cluster in which $\mathbf{x}_i$ is located, and average this quantity over all nodes to yield the final plotted value. Figure 7.11 illustrates this for the *eurovis* dataset. Nodes are here colored to show the identity of the clusters computed by $k$-means from this dataset. We see that the node colors in the projection using the *proj-force* method (Fig. 7.11b) seem to match the nearest-neighbors of their respective points better than when using the *force* method (Fig. 7.11a). This is precisely the property that *nhit* aims to capture. Figure 7.12(a,c,e) shows that preconditioning the force-directed layout by a projection (*proj-force*) yields consistently higher *nhit* values than when using a standard force-directed layout (*force*).

Several additional global observations are due to the quality analysis presented in Fig. 7.12, as follows. First, we note that the maximal values for both *npres* and *nhit* for our studied datasets, respectively $npres = 0.425$ and $nhit = 0.83$, are not very large, given that the absolute maxima of both metrics is one. This reflects the inherent difficulty of mapping the connectivity patterns of the three considered relational datasets to a two-dimensional space. However, as stated earlier, we obtain higher values for both metrics when using our proposed *proj-force* method than when using the standard *force* spring-embedder method. As such, we conclude that the projection-based preconditioning of force-directed methods is beneficial in terms of increasing the quality of the final visualization. Secondly, we see how both *npres* and *nhit* depend on the number $k$ of considered nearest neighbors. This phenomenon is expected – as discussed in more detail in Chapter 4, preserving smaller neighborhoods is, in general, much easier than preserving large neighborhoods. We also note that, to get a more detailed comparison and understanding of neighborhood preservation issues involved with the *force* and *proj-force* methods, we could employ here the fine-grained set-difference (Sec. 4.1.3) and sequence-difference (Sec. 4.1.4) views. However, our current purpose is to obtain a *global* understanding of which of the two considered graph-drawing methods, *force* and *proj-force*, has a higher quality, rather than interpreting the fine details and errors of a given projection. As such, we prefer for this task the aggregated error plots shown in Fig. 7.12. Finally, we note that we obtained similar *npres* and *nhit* curves, and observed a similar quality advantage of the *proj-force vs* the *force* method, when using other projection techniques than IDMAP, *e.g.*, LSP [139]. This increases our confidence when we state that multidimensional projections are useful precoditioners to force-directed layouts.

To get a better insight in the perceived advantage of *proj-force* as compared to *force*,



(a)

(b)

(c)

(d)

(e)

(f)

Figure 7.12: Quality plots for both *force* and *proj-force* layouts. Top row: *eurovis* dataset; middle row: *vis* dataset; bottom row: *agric* dataset. Left column: neighborhood hit metric. Right column: neighborhood preservation metric.

we computed, for each pair of plots (*i.e.*, $npres_{force}$ *vs* $nhit_{force}$ and $npres_{proj-force}$ *vs* $nhit_{proj-force}$), the significance of their differences. For this, we used a simple statistical $t$-test. The resulting $p$-values are shown in Table 7.3. They demonstrate the growing significance of the advantages of performing a multidimensional projection as an initial step to a force-directed layout, as the number of nodes grows. Table 7.3 also shows our third and final quality metric used to compate the *force* and *proj-force* methods – the number of edge-crossings counted for a given layout. As well known from numerous analyses, edge-crossings are undesirable artifacts in a graph drawing, as they impair the easy and reliable reading of the connectivity information conveyed by a node-link visualization [125, 181]. As such, graph visualizations which minimize such edge crossings are preferred. In Tab. 7.3, we see that the number of edge-crossings is (significantly) lower for the *proj-force* method as compared to the *force* method. This is an additional argument in favor of our proposal to precondition force-based layouts by multidimensional projections.

Table 7.3: Aggregated comparison values for *proj-force* and *force* plots.

| **Dataset** | **nhit p-value** | **npres p-value** | **edge crossings proj-force** | **edge crossings force** | **number of clusters** |
|---|---|---|---|---|---|
| *eurovis* | 0.25 | 0.16 | 45,600 | 54,500 | 10 |
| *vis* | 0.17 | 0.17 | 1,620,000 | 2,900,000 | 10 |
| *agric* | 0.0013 | 0.029 | 4,000,000 | 7,900,000 | 15 |

### 7.3.3 Connectivity-based projections: Neighborhood preservation

Section 7.3.2 has presented an aggregated quantitative analysis of the added-value of preconditioning force-based graph layouts by multidimensional projections. As mentioned in that discussion, these aggregated views fulfill their aim quite well – that is, showing whether and how much one gains from using projections before force-based layouts.

A separate question is: How to provide fine-grained insight in the quality issues involved in our projection-based methods used for the visualization of networks? In other words: How can/should we adapt the detailed neighborhood preservation views, presented in Chapter 4 for multidimensional datasets, to handle multivariate relational datasets? To answer this question, we chose to adapt the set difference view described in Sec. 4.1.3 to the context of networks. In detail, to do this, we need to provide a definition of distances in the high-dimensional space between observations, which are used to compute neighbor sets (Eqn. 4.3). For this, we proposed to use in our network context the shortest-path distance matrix introduced in Sec. 7.2.1.

While the shortest-path distance matrix can, technically, be used without modification to compute the set-difference view, in terms of finding the high-dimensional $k$-neighborhood of any node, there are several special cases where this idea needs to be adapted, as follows:

**In many not strongly-connected graphs, some nodes may not have $k$ neighbors.** In this case, a method that determines a $k$-neighborhood of a node $\mathbf{x}_i$ may include as

$k$-neighbors of $\mathbf{x}_i$, and for certain $k$ values, nodes that are not connected to $\mathbf{x}_i$. This is due to the fact that, in our shortest-paths distance matrix, distances between disconnected nodes are all set to the fixed value $\max_{i,j} P$, where $P$ is the maximal path length between any two nodes in the input graph (Sec. 7.2.1). To handle this, we adapt the global user-provided value of $k$ nearest neighbors in a local way: Given a value of $k$, a node $\mathbf{x}_i$ will have $k' < k$ neighbors considered in its neighborhood-preservation analyses, if the connectivity structure of the input graph states that $\mathbf{x}_i$ has only $k'$ nodes connected to it. The same value $k'$ is used next to determine the number of neighbors of the projection of $\mathbf{x}_i$ in 2D when computing, for example, the neighborhood-preservation *npres* quality metric (Sec. 7.3.2).

**In a shortest-paths distance matrix of a network with discrete edge weights, several $k$-neighbors of a node $\mathbf{x}_i$ may have the exact same distance to $\mathbf{x}_i$.** The problem here is that, when computing the $k$-neighborhood of $\mathbf{x}_i$ given by this distance matrix, the search for neighbors may finish with an arbitrary neighbor that has precisely the same distance to $\mathbf{x}_i$ than several other neighbors. Since all these neighbors have precisely the same distance to $\mathbf{x}_i$, we do not know *a priori* which to include in the $k$ neighbors of $\mathbf{x}_i$. This is a well-known 'tie' situation encountered also when one searches for $k$ nearest neighbors in Euclidean distance spaces [6]. To solve this problem, we include in the $k$-neighborhood of $\mathbf{x}_i$ all neighbors having equal distances to $\mathbf{x}_i$ equal to the distance of the $k^{th}$ neighbor. Thereby we solve the aforementioned ties by including all of them in the $k$-neighborhood of $\mathbf{x}_i$. Note that the problem of ties is also present in the original set-difference view presented in Sec. 4.1.3, albeit to a lower extent, since the likelihood that many high-dimensional Euclidean distances between observations are precisely equal is far less than in the case of using connectivity distance matrices with discrete edge weights.

With these two adaptations, the set difference view in Sec. 4.1.3 can be applied to connectivity-based projections in order to compare the $k$-neighborhoods of points in the original (network) space, given by the shortest-paths distance matrix, and those implied by the resulting network drawing in 2D. Figure 7.13 presents the results of such an analysis for the *eurovis* dataset – one showing the full paper-author network (Figs. 7.13a,c) and the second one showing the largest connected component (Figs. 7.13b,d), for the two types of distance matrix proposed in Sec. 7.2.1, *i.e.*, the modified-adjacency and the shortest-paths distance matrices. The chosen projection method was ISOMAP [184]. The number of considered nearest-neighbors $k$ was set to 10% of the total number of nodes in each graph, *i.e.*, 128 nodes for the full graph, and 72 nodes for the main component, respectively.

The layouts resulting from projections of the full *eurovis* graph (Figs. 7.13a,c) are very different from each other, while the layouts resulting from projections of the main component (Figs. 7.13b,d) are more similar. Nodes in these figures are colored by the neighborhood-preservation error, using a blue (low) to red (high) colormap. This confirms the large impact that sets of small disconnected sub-graphs have in the final results of the projections of networks, in line with earlier findings discussed in Figs. 7.9 and 7.10 and related text. In Figure 7.13a it is possible to discern that these small isolated subgraphs (dark blue component to the right of Fig. 7.13a) all show a very low

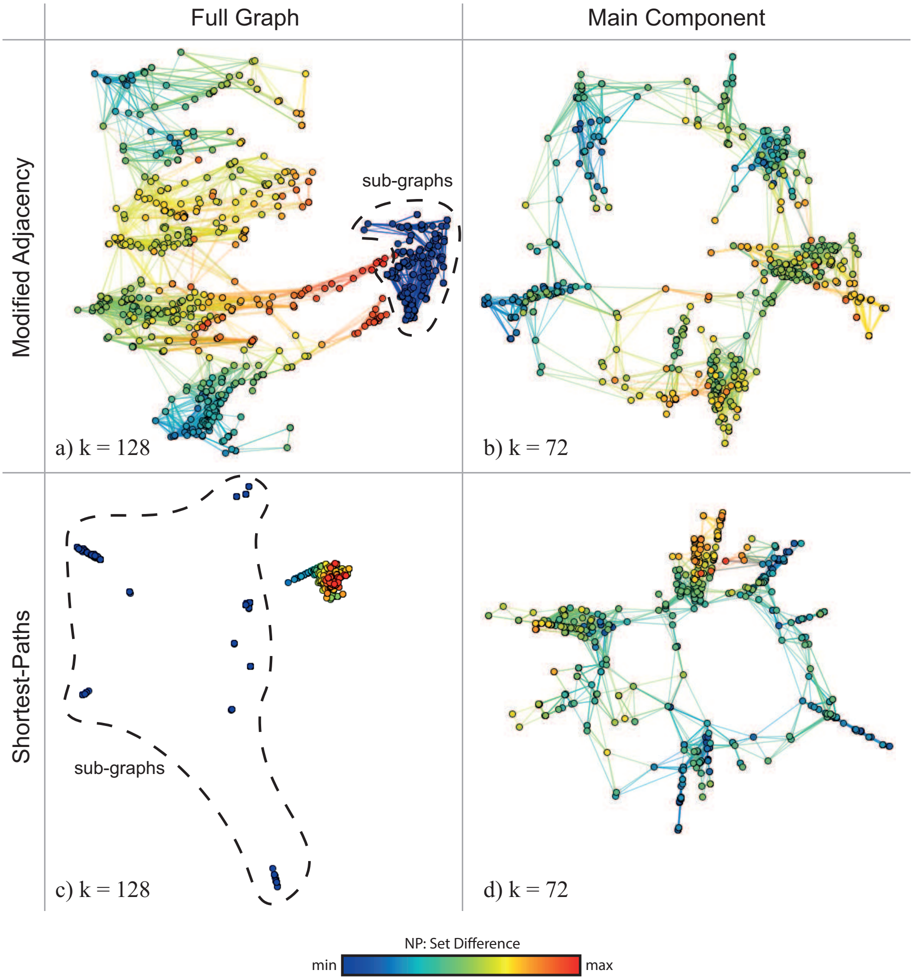


Figure 7.13: Analysis of neighborhood preservation using the set-difference view for two versions of the *eurovis* dataset. (a,c) Full network. (b,d) Main connected component. The graph drawings were created by (a,b) the modified adjacency distance matrix; and (c,d) the shortest-paths distance matrix of the considered networks.

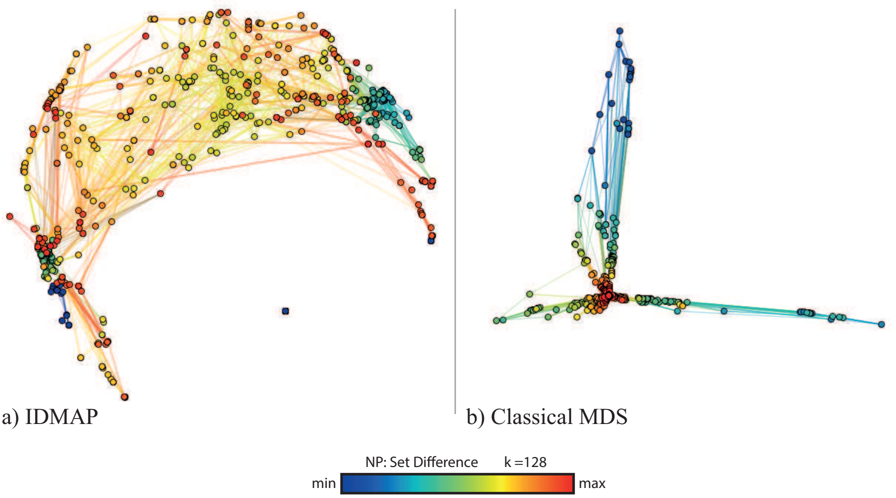
neighborhood preservation-error, as illustrated by the node's colors, while the highest errors are located in edges that reach this subset of isolated graphs. These errors are very likely due to the close positions of these nodes in the 2D layout, while the original (connectivity-based) distances are quite large, as this component is disconnected from the remainder of the graph. The remainder of the network (large component in the center and left of Fig. 7.13a) is roughly divided in horizontal bands of points with similar levels of neighborhood-preservation errors. This reflects the tendency of the modified adjacency distance matrix to group immediate-neighbor nodes with less focus on the global distances. Two groups of nodes with smaller errors are positioned on the top and the bottom of the layout (green-cyan nodes in Fig. 7.13a), while the central area is populated with nodes having middle-to-high errors. This indicates that these central nodes (authors and papers) have a more confusing neighborhood structure on the original graph, in terms of having many connections with several node groups, which are hard to embed in two dimensions.

The shortest-paths-based layout of the full graph (Fig. 7.13c) is not very informative: The large main component of the graph has collapsed into a small lump of nodes (compact component showing warm colors to the top-right in Fig. 7.13c). All other other smaller sub-graphs have a low neighborhood preservation error and are spread around the 2D space. This shows that, for this specific projection, using distances from all pairs of points might induce a process of distortion of inter-point similarities inside the main component and the inter-group dissimilarities between all sub-graphs. On the other hand, the similar layout and distribution of neighborhood-preservation of the two projections of the main connected component of the graph (Figs. 7.13b,d) are an interesting indication that the original structure of this large group is well captured, showing a few important sub-groups with high connectivity, linked by a few nodes representing their research collaborations. Given of this characteristic of this part of the graph, these sub-groups and their structure are well preserved in the 2D layout, regardless of using the modified adjacency or shortest-paths distance matrices.

Besides the variations caused by the structure of the network and the input distance matrix used in the projection process, different projection methods will also generate different layouts depending on how their inner algorithm works. In order to better clarify the effects of the projection method on the layout of the networks and their correspondent neighborhood preservation errors, in Fig. 7.14 we show two extra projections of the full graph, using the modified adjacency distance matrix as input (in line with Fig. 7.13a). The multidimensional projection methods used here are IDMAP [122] and classical multidimensional scaling (MDS) [17].

As observed in Fig. 7.13, the set of small isolated sub-graphs behaves, again, very differently for each projection technique. With the IDMAP projection (Fig. 7.14a), these graph components are collapsed into a single 2D location. This is a common pattern for this projection method, as shown, for example, in Figs. 7.1 and 7.9. While it may seem that these sub-graphs are being neglected by the projection, in favor of laying out the largest group, the wide empty circular area that is kept around them suggests that it is not so: Their inner organization of these small components is not shown, but their effect on, and distance to, the other nodes, which is due to the high distances in the distance matrix between disconnected nodes, is clearly reflected. Inside the main connected group in Fig. 7.14a, patterns between the nodes are hard to discern, as most of these nodes are regularly distributed in the 2D embedding space and also show mid-to-high neighborhood preservation error. Yet, these values are quite different from the errors shown for the two smaller groups on the left and right sides of the projection, which have lower neighborhood-preservation errors.

Figure 7.14b shows one final comparison of our *eurovis* dataset, using this time a layout constructed by the classical MDS projection method [17]. In this case, in a similar way to Fig. 7.13c, most nodes of the main component were collapsed into a small 2D



a) IDMAP

b) Classical MDS

Figure 7.14: Analysis of neighborhood preservation errors using the set difference view for the *eurovis* dataset, using different projection techniques. The network connectivity is encoded by using the modified adjacency distance matrix.

area, making it hard to discern their inner organization. The neighborhood preservation error in this central area is high (red cluster of nodes in Fig. 7.14b), hinting at a possibly inaccurate layout. Three groups of nodes, however, were positioned in very distinct groups in long 'arms' stretching from the central tight group. These arms also show lower neighborhood preservation errors than the central group. This is an indication that the MDS projection method, when used with a modified adjacency distance matrix, considerably favors the separation of well-connected sub-groups instead of showing the often more complex layout of tightly-connected node groups. Another characteristic of the layout of Fig. 7.14b is that the smaller sub-graphs are not immediately visible, as with most other layouts, but are mixed with the rest of the connected graph. This tells that, for this particular projection method, the large dissimilarities between disconnected nodes are not taken into account as much as, for instance, the IDMAP projection illustrated in Fig. 7.14a.

### 7.3.4 Attribute-based projections: Multivariate software networks

Apart from social networks, such as those created by considering papers and authors discussed in the previous sections, software systems offer another rich field for the exploration of multivariate attributed networks. Indeed, software systems can be described by graphs, or networks, where nodes are software entities, *e.g.* functions, classes, files, packages, and subsystems; and relations describe the interaction of such software entities, such as dependencies, call relations, data flows, include relations, and code duplication [45]. Nodes have a wealth of attributes, such as name, signatures, and software quality metrics [107]. Altogether, this delivers us rich multivariate attributed graphs which we can explore to understand the software systems at hand from multiple perspectives.

Below we describe several scenarios used to explore the multivariate network structure of software datasets by our proposed multidimensional projection methods introduced in Sec. 7.2.1). The procedure for creating such networks from software projects is outlined below.

1. **Obtaining the source code:** The source code of the studied software projects was obtained from *github* [65], a popular source-code repository for open-source software projects. For analysis, we selected several popular projects, having many commits, active developers, and clones. For each project, we selected to analyse its most recently updated revision in the project's master branch. We selected projects written in the C++ programming language, as our software-analysis tools, which were used next to extract relations and attributes from source code, support this language well.
2. **Attribute extraction:** To characterize the software entities in each project, we need to extract a number of attributes for each such element. For these, we used 37 code quality metrics frequently used in software maintenance, including object-oriented measures such as structural complexity, coupling, cohesion, and descriptive measures such as total lines of code, total number of methods and attributes [33, 10, 107]. We automatically extracted these metrics from the downloaded source code of the studied projects using Analizo [185], an open-source tool for source code static analysis. Other static analysers, *e.g.* [183], can be used equally well. The result of this analysis is a set of software entities, each being described by a 37-dimensional vector of real-valued (quantitative) attributes.
3. **Relation extraction:** The software entities identified in our studied projects are connected by dependency relationships which are explicitly defined in source code in terms of method calls – an entity $A$ *depends* on an entity $B$ when at least one method of $A$ calls at least one method of $B$. The resulting dataset is typically known as a *call graph* in program comprehension [45]. For simplicity, we chose to consider edges as being undirected. The weight, or strength, of an edge connecting two entities $A$ and $B$ equals the sum of the number of methods from $A$ calling methods from $B$ with the number of methods from $B$ calling methods from $A$. In our terminology, this is a discrete edge attribute.

Merging these two features – multiple attributes per node from extracted software metrics and connections between nodes – we construct multivariate software network datasets that can be described either by the characteristics of their individual nodes, or the relationships between their nodes, or both at the same time, depending on the goals of the analysis. We next detail several visual exploration applications of such datasets based on our projection methods.

The dataset discussed in this section was extracted from the source code of the software project *caffe*, a deep learning framework developed mainly by the Berkeley Vision and Learning Center (BVLC) at the University of Berkeley, California [93].

**Attribute-only projection:** Considering first the multivariate nature of the dataset, its many attributes (metrics) mapped on each element make it possible to construct a first visualization of this dataset, using a multidimensional projection method that positions the elements (nodes) in 2D based on distances that reflect their attributes' similarities. Figure 7.15 shows the result of applying the LSP projection method [139] on this dataset, with no connectivity information considered during the projection process.

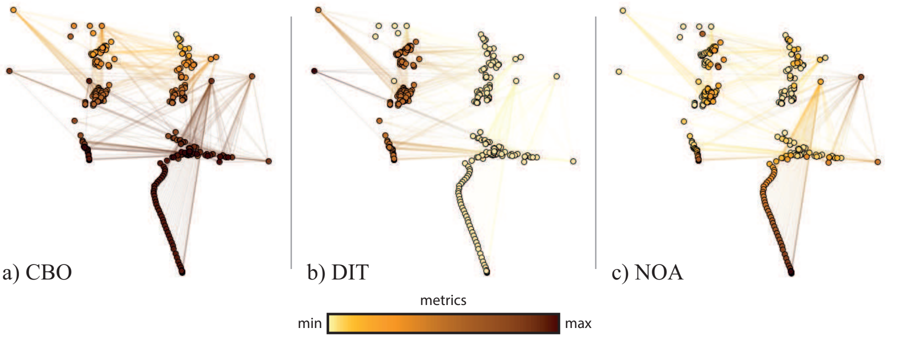


Figure 7.15: Attribute-based projection of *caffe* dataset, using the LSP projection technique, with three color-coded metrics: (a) Coupling between objects (CBO); (b) Depth of inheritance tree (DIT); and (c) number of Attributes (NOA) [33, 10, 107].

To get more insight in how attributes determine the projection, we color the projected nodes in Fig. 7.15 by the value of three different selected metrics: (i) coupling between objects (CBO), which counts the unique number of other types (classes and interfaces) that are related to a node through method calls, method parameter types, return types, thrown exceptions, and accessed fields [33]; (ii) depth of inheritance Tree (DIT), which measures the maximum length of the class-inheritance path from a given (class) node to the root of its inheritance tree [33]; and (iii) the number of attributes, which gives, for any node, the number of data attributes our static analysis computed for that node.

The three color-coded metrics shown in Fig. 7.15a–c lead to various insights and interpretations of the resulting projection layout. In Figure 7.15a, we see that the values of the CBO metric are distributed in increasing order from top to bottom (with only a few outliers), resulting in a color gradient along the $y$ axis of the 2D projection space. In other words, the CBO metric increases gradually from bottom to top in this projection. In Figure 7.15b, the gap separating the two vertical bands of points located at the left, respectively right, of the projection, appears to be explained by the DIT metric, which takes only three possible values in the dataset – low for the right group of points (yellow), high for the left group of points (orange), and very low for a single outlier point located to the extreme left of the projection (brown). Figure 7.15c shows that the NOA metric is related to the formation of a tail-like structure in the lower part of the projection; all points in this structure show high CBO and low DIT values, but they vary significantly with respect to their number of attributes (NOA) values.

**Connectivity-only projection:** A different, but very common, way of looking at the structure of a software project is to visualize it as a dependency graph, where nodes can be files, classes, modules, methods, or any other structural element of the programming language used; and edges represent connections between these elements, *e.g.* dependencies, communication paths, or containment relations. Using the connectivity-based projection techniques described in Sec. 7.3.1, Figure 7.16a shows an LSP projection created from the shortest-path distance matrix of the *caffe* dependency graph. In order to contrast this new layout with the one presented in Fig. 7.15, we replicate it in Fig. 7.16b with a new visual encoding: The three metrics displayed in separate subfigures in Fig. 7.15 (CBO, DIT, and NOA) and are also shown by Fig. 7.16; however, we now attempt to let one visualize these three metrics simultaneously, and therefore encode them into three separate visual properties of the network nodes, as follows: (i) Since the DIT metric only assumes three values in the entire dataset (0, 1 and 2), we encode it by node-glyph shape, by using squares, circles, and triangles for the three values 0, 1, and 2 respectively. (ii) The values of the NOA metric, which were shown earlier to vary mostly in a specific region – the tail-like formation at the bottom of the projection in Fig. 7.15c – are now encoded by glyph sizes; and (iii) the CBO quantitative metric is color-coded by a heat colormap.

In Figure 7.16a, the glyph properties (colors, shapes, and sizes) appear to have little relation to the positions of the points, being randomly distributed throughout the entire layout. This is expected, since the LSP projection used here uses purely connectivity (shortest-path) information and no attribute information. The high-DIT-valued nodes, however, are mostly distributed in a group of points in the lower-left part of the main connected component in Fig. 7.16a, along with most of the high-CBO-valued nodes. The tight grouping of nodes with these two attributes may be explained by two observations: First, the way the CBO metric is computed is also related to the dependencies (relations) of the network, since method calls, which we consider here as edges between nodes, are one of the relationships that are counted for the metric. This means that high-CBO-valued nodes are also highly-connected nodes in the network; hence, a connectivity-based projection will tightly group such nodes. Secondly, most of the high-DIT-valued nodes that form the tail-like structure in marked in Fig. 7.16b are connected to a single other node (*gtest* in Fig. 7.16b). This node is also high-CBO-valued and is positioned inside the same tight group of nodes in the lower-left part of the main component marked in Fig. 7.16a.

In order to understand better the shared characteristics of the connectivity-based and attribute-based projections of the *caffe* dataset, we select the same group of high-CBO-valued nodes in both Figs. 7.16c,d, which show these two projections, and next manually browse their elements to investigate their similarities. By brushing this group of points in both images at the same time, following a linked-view metaphor, several insights appear: (i) The tail-like formation derived from the NOA values is indeed well-hidden in the connectivity view (Fig. 7.16a); (ii) a few nodes, such as the ones marked in Fig. 7.16d, have a very high degree or number of edges; however, since the connectivity view favors placing nodes close to their connected neighbors, this is hard to see behind the clutter; and (iii) the strongy-connected and high-CBO-valued group of points marked in our selection is related to testing code, and not the separate ‘production-grade’ code of the

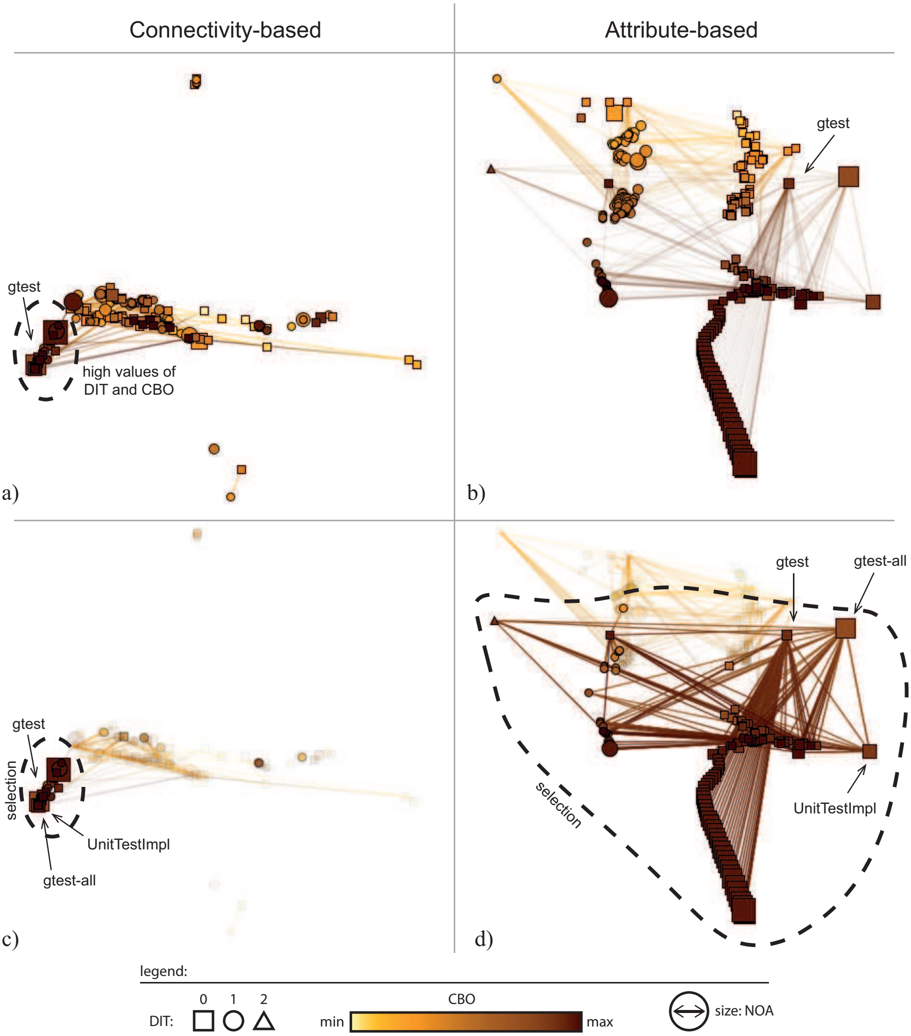


Figure 7.16: Projections of the *caffe* dataset, using the LSP projection technique, with attribute values encoded by node glyph sizes and colors. (a,c) Projection based on the shortest-path distance matrix only; (b,d) Projection based on the attribute values only. Bottom row (b,d) focuses the visualization on two selections of nodes, and thereby renders nodes outside these selections as half-transparent.

*caffe* software system.

**Attribute-and-connectivity projection:** As it can be noticed by the previous analysis of Figs. 7.15 and 7.16, the two different views of the *caffe* dataset generated by using either attributes or connectivity information are each useful in their own way, showing insights that relate to different aspects of the system under investigation. However, they both also have their limitations. The attribute-based view does nothing to ensure that the resulting 2D layout is coherent with the dependency structure (relations) of the network; this aspect is prominent when seeing the long and multiply-crossing edges in Figs. 7.15 and 7.16b,d. In contrast, the connectivity-based projection groups nodes by their dependencies (relations), but does little to facilitate the examination of nodes with different attribute values; such nodes are often collapsed into small areas which are too compact for visual exploration, or alternatively they are spread randomly over the projection. To deal with situations where both attributes and connectivities are equally important for the analysis at hand, which is usually the case when investigating software systems, we can use a projection based on the combination of attribute-based and connectivity-based distance matrices introduced in Sec. 7.2.2. The results of using this approach with the *caffe* dataset are shown in Fig. 7.17.

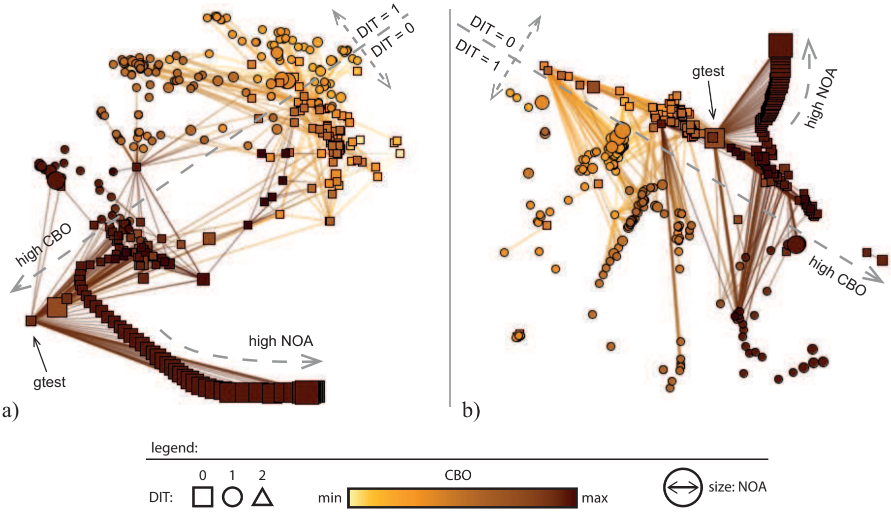


Figure 7.17: Two LSP projections of the *caffe* dataset, using combinations of the attribute-based and the connectivity-based distance matrices. Connectivity information is represented using (a) the modified adjacency distance matrix, and (b) the shortest-paths distance matrix.

Following patterns already observed in previous examples, *e.g.* Figs. 7.6 and 7.13), the projection that uses the modified adjacency distance matrix (Fig. 7.17a) has a tendency of better using the available screen space, spreading the nodes more evenly. In contrast, the projection that uses the shortest-paths distance matrix (Fig. 7.17b) is more strict and tends to group nodes more strongly. Due to the effects of the attributes, however, neither of the above two projections show the high level of compactness of Figs. 7.16a–c. This makes it easier to discern structural patterns among the nodes, such as the high-degree nodes separated from the main group in Fig. 7.17b.

When looking at the visual encodings of the three metrics in Figs. 7.17a–b, we rediscover the patterns previously observed in the attribute-only projection (Fig. 7.15), even though our new projection in Fig. 7.17 combines attributes and connectivity information. This stands our in contrast to the connectivity-only projection (Fig. 7.16). The gradient of colors that depict the values of the CBO metric is again visible, although in a different direction in each projection. Again, we discover the group of high-valued CBO nodes close to each other; however, in contrast to the attribute-only and connectivity-only projections, we now obtain a more spread-out layout that allows for node interconnections to be explored more easily.

The previously-noticed tail-like structure (Fig. 7.15) is now visible in both layouts (Figs. 7.17a–b). This reflects the fact that, even though these nodes are very similar in terms of connectivities, their values of the NOA metric increase considerably and must be differentiated in the view. The edges that emerge from these nodes, however, are now shorter and show less crossings, due to the better positioning of the *gtest* node in relation to the group.

A final insight is found by looking at a direction perpendicular to the direction of the color gradient showing the CBO metric. In this perpendicular direction, we see how the DIT-metric values are distributed into different sides of the projection – see annotations $DIT = 0$ and $DIT = 1$ in Figs. 7.17a–b – in a similar manner to the effect shown in Fig. 7.15. However, in this case, instead of obtaining completely isolated groups with different DIT values, we have a rough division of the projection in two halves which still maintains the coherence given by groups of well-connected nodes. The separation between the different DIT valued nodes occur *inside* each node-group; this effectively combines both the insights that nodes in these groups are (1) similar, in terms of connectivity; but (2) different, in terms of DIT attribute values.

### 7.3.5 Multivariate software networks: Quality analysis

The next application is based on the source code of the *Bitcoin* project [126], a digital currency trading platform implemented as a decentralized peer-to-peer network. We use the multivariate software network of *Bitcoin*, extracted from the project's source code using the process described in Sec. 7.3.4, to investigate the effects of using combined connectivity-and-attribute distance matrices, with different projections methods, on the neighborhood preservation of the nodes in the final 2D layout. For a fine-grained local quality analysis, we employ again the set difference view adapted to multivariate networks described in Sec. 7.3.3.

Figure 7.18 shows the results of using different types of distance matrices to generate projections, using the LSP technique, of the main connected component of the *Bitcoin* dataset, which we next call *Bitcoin-Main*. The colors of the nodes represent their neighborhood-preservation error, computed by the network-adapted set-difference metric, using a colormap ranging from blue (low error) to red (high error).

In the images presented in Fig. 7.18, each row contains projections created with one

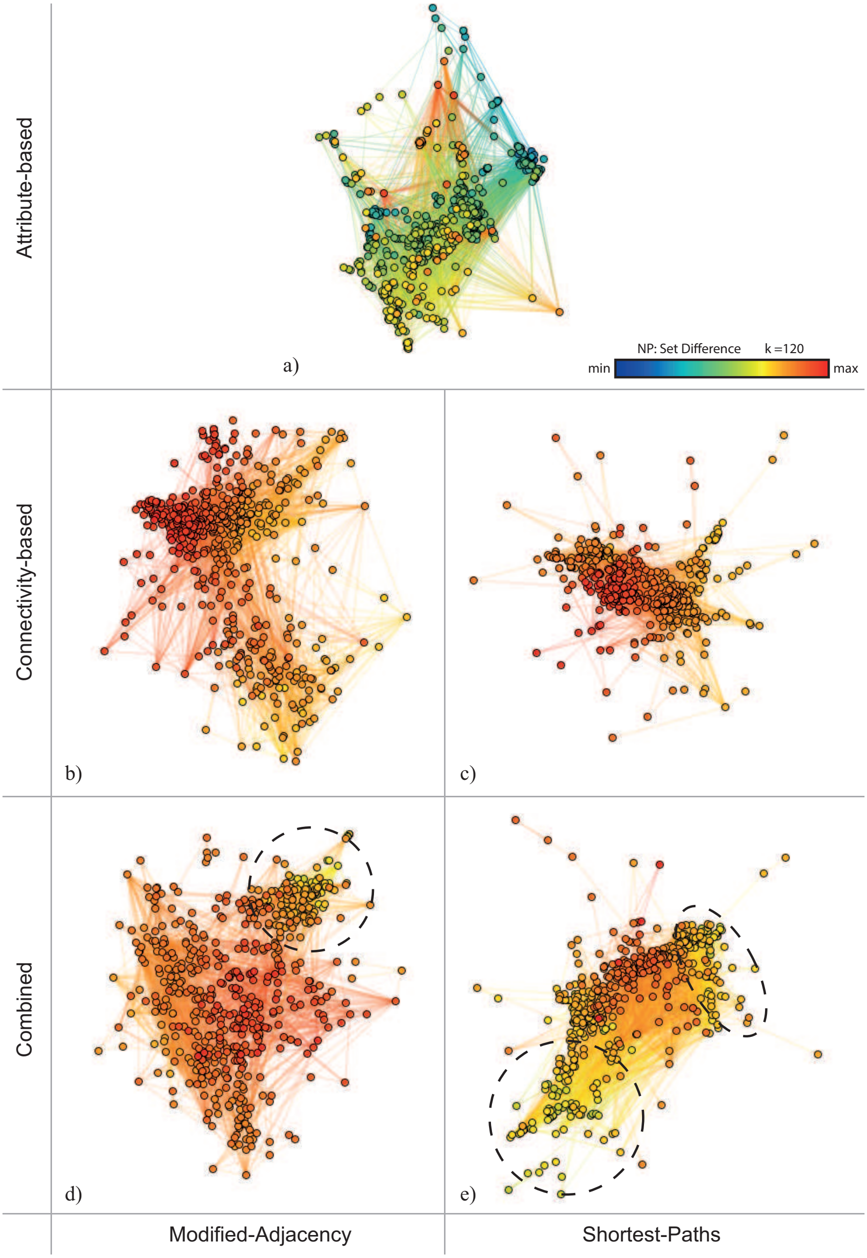


Figure 7.18: Neighborhood preservation error (set difference view) analysis of the *Bitcoin-Main* dataset, using LSP projections with different inputs: (a) Attributes only; (b) Connectivity only (modified adjacency matrix); (c) Connectivity only (shortest-paths matrix); (d) Combined attributes and modified adjacency matrix; (e) Combined attributes and shortest-paths matrix.

of the three different approaches for the visualization of multivariate networks presented in this chapter, which means that the *original* high-dimensional distances used for creating the projections are different. For this reason, when computing the neighborhood preservation errors of the projections of each row, the input distance matrices used were different, as follows: a distance matrix generated from Euclidean distances between the nodes' attributes (Fig. 7.18a); the shortest-path distance matrix, similar to the approaches described in Sec. 7.3.3 (Figs. 7.18b,c); and a distance matrix combining the Euclidean distance between node attributes and their shortest-path distance in the network (Figs. 7.18d,e). The number $k$ of nearest neighbors used when computing the neighborhood preservation errors was set to $120$, which represents $20\%$ of the total number of nodes. This was raised from the initial $10\%$ used in previous applications (Sec. 7.3.3), due to the even higher overall errors observed for lower $k$ values.

A first interesting insight from Fig. 7.18 is that, besides the attribute-based projection (Fig. 7.18a), all other shown projections, which use connectivity as a factor, display relatively high neighborhood preservation errors (Figs. 7.18b–e). The errors are especially large for the projections that use only connectivity information (Figs. 7.18b,c). While the projections created from combined distances also show a similar trend (mid-to-high errors), it's possible to see that these contain several relatively large areas (marked in Figs. 7.18d,e) with nodes with lower errors. These areas are localized in the periphery of the resulting layout, with more central nodes showing higher errors. This shows that these projections respect the original structure of the input dataset (in terms of relations *and* attributes) better than the connectivity-based projections.

If we compare the above observations with the completely different picture shown in Fig. 7.18a, we see that the attribute-based projection shows mostly mid-to-low error values over basically the entire layout. Hence, we conclude then that using LSP for connectivity-based projections generates tightly-packed layouts (confirming the features seen previously *e.g.* in Figs. 7.16a,c), which do not preserve local neighborhoods very well. In contrast, the quality of LSP with attribute-based (or combined) distance matrices is generally better. To test this observation, we select the set-up shown in Fig. 7.18e – combined attribute-and-shortest-paths based distance matrices – and use it to generate projections with two other different previously-used methods: IDMAP and ISOMAP. The results of the neighborhood preservation analysis, for the same value of $k = 120$, are shown in Fig. 7.19.

The IDMAP projection of the combined distance matrix (Fig. 7.19a) also shows a tight grouping of the nodes, similarly to what was observed with the LSP projection, but with a clearly different error distribution: Instead of lower errors on the periphery of the layout and higher errors towards the center, as the case of LSP was, IDMAP shows exactly the opposite – central nodes have the lowest errors seen in any of the combined attribute-connectivity projections of the *Bitcoin-Main* dataset. While the nodes of the ISOMAP projection in Fig. 7.19b do not reach the same low-error values of the IDMAP example (Fig. 7.19a), their errors are still overall lower than those shown for all LSP-based layouts in Fig. 7.18. The ISOMAP projection also has one other interesting characteristic: The error values show a roughly linear gradient: At the right of Fig. 7.19b, nodes are more sparsely distributed and their errors are lower than to the left of the

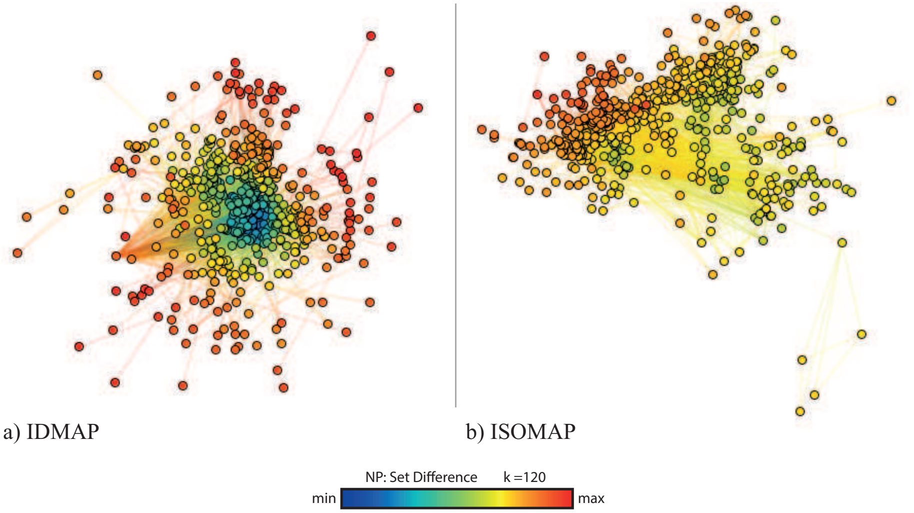


Figure 7.19: Neighborhood preservation analysis of the *Bitcoin-Main* dataset, using the IDMAP and ISOMAP projection techniques. Nodes are encoded by a combined attributes-and-shortest-paths distance matrix.

same figure, where nodes are more tightly-grouped and their errors are higher. Similarly to LSP, but contrary to IDMAP, the tighter node groups of the ISOMAP projection show more error, but at the same time the nodes with lower errors and their neighborhoods are more visible. These observations positively corroborate our conclusion that LSP, while working well with attributes, is arguably not a good candidate for connectivity-based projections.

## 7.4 Discussion

Several aspects of our proposal to use multidimensional projections to create 2D visualizations of attributed networks are important to discuss, as follows.

**Distance-based projections:** As already mentioned in Sec. 7.2.1, all projections that were used in our construction of visualizations of attributed networks are essentially multidimensional-scaling-type (MDS-type) projections. As explained with several occasions in this thesis, a second class of projections exist, which accepts multidimensional observations rather than distance matrices, *e.g.*, LAMP [95]. Although we have experimented with using such projections for constructing 2D visualizations of attributed networks, our results so far have been markedly of less quality than when using distance-based projections. The key problem here seems to be related to the design of a *meaningful* multidimensional attribute vector for observations (nodes) which encodes both their attribute value *and* their connectivity pattern. Technically, many schemes can be created which translate relational information into attributes in an algorithmically consistent way. However, our experiments so far have shown suboptimal results as compared to projections using distance-based matrices. As such, the usage of projections that do not use distance matrices to create network visualizations is still an open research topic.

**Mixing connectivity and attribute information:** We have shown that we can create projections based purely on a network's connectivity information, its node attributes, or a (weighted) mix of the two. Not unexpectedly, mixing connectivity and attribute information creates visualizations which are, in a qualitative sense, an 'interpolation' of the visualizations using connectivity or attributes only. The added-value of all these different visualization styles is highly dependent on the use-case at hand – there are several envisageable scenarios where a connectivity-only, an attribute-only, or a combined visualization are preferred. An interesting area for further research, however, is exploring how one can control the *visual* differences of such network layouts in terms of the used weights for the construction of the underlying distance matrices. For instance, we would find it useful if it were possible for users to specify how many units of two-dimensional (embedding) space were to be affected by, *e.g.*, relations, attributes, or even a specific selection of attributes.

**Projection space:** As illustrated by our several examples in this chapter, we can create projections which differ visually in significant ways by varying several parameters, such as the kind of projection technique, type of connectivity-distance matrix, weighting of attribute-distance matrix with the connectivity-distance matrix, and selection of attributes being used in the attribute-distance matrix. The richness of the exploration space opened by multidimensional projections is not surprising – we have discussed a similar phenomenon for the use of projections to visualize non-relational multidimensional datasets in Chapter 3. As such, and in line with the discussion and findings outlined in Chapter 3, we note that our exploration of the usage of projections to visualize multivariate graphs discussed in this chapter is only a (very) limited sampling of the entire space of possible parameter possibilities. A more thorough study of all combinations of parameter values is of clear added-value in the sense of understanding the working of multidimensional projections. Let us also note that this large space of parameters, or possibilities, is, at a higher level, identical to the well-known space of possibilities for creating significantly different drawings from the same graph. Just as we miss a comprehensive exploration of the possibilities of drawing a graph, we advocate the need of better exploration of possibilities of using multidimensional projections for drawing attributed graphs.

**Projection quality:** In this chapter, we have shown that aggregated neighborhood-preservation errors (Sec. 7.3.2) and the more detailed set-difference views (Sec. 7.3.5) can be effectively used to explore projection errors incurred when drawing multivariate networks with multidimensional projection techniques. However, several other ways to quantify and visualize projection errors exist, as described in our earlier work on distance-based errors (Chapter 3) and the sequence-difference views for neighborhood preservation errors (Section 4.1.4). Technically speaking, adapting these error metrics and visualization to work for multivariate networks is straightforward. We leave the exploration of the insights delivered by these metrics in the overall quality of multivariate network layouts constructed with projections as a topic for future work.

**Visual Presentation:** In Chapters 3 and 4), we have shown several examples of networks or graphs which use edge bundling techniques to spatially group edges connecting closely placed endpoints (nodes). As explained in the respective chapters, edge bundling creates simpler and less cluttered visualizations of large graphs, and thereby helps one detect important connectivity patterns in the graph. In contrast, the network visualizations shown in this chapter were created by using the classical node-link metaphor for graph drawing. That is, nodes are placed using multidimensional projections, just as in the applications described in Chapters 3 and 4). However, instead of drawing edge bundles, we chose to draw edges as straight lines. This visualization design decision was taken in order to focus the attention of the reader, when examining the resulting images, on the specific details of *node placement*, which are the key contribution of this chapter. Additionally, steaight-line node-link drawings allow an easier discussion and comparison of the impact of the different proposed techniques for creating graph layouts, while edge bundling visualizations tend to focus the attention of the observer mainly on the *connection patterns* represented by the edges.

However, the above does not mean that edge bundling cannot be applied to, and is not useful for the visual exploration of, graph layouts constructed by the projection techniques presented in this chapter. To illustrate this, Figure 7.20 shows an alternative visualization of the graph layouts presented earlier in Fig. 7.6, by using the edge bundling technique described in [182] and used earlier in Chapter 3. Node colors and sizes encode the cluster identities and betweenness-centrality measure respectively, as in Fig. 7.6. Edges are rendered with alpha blending. As such, less dense bundles appear as half transparent (light gray), while dense bundles containing many edges appear darker. As visible from Fig. 7.20, edge bundling is very effective for showing the coarse-scale main connectivity patterns in the depicted graph, and creates less cluttering than when using straight-line drawing. In the same time, individual edges in the graph cannot be easily followed in the bundled visualization – a well-known limitation of edge bundling techniques.

## 7.5 Conclusions

In this chapter, we have explored the use of multidimensional projections for the construction of visualizations of multivariate attributed graphs, of networks. The core of our proposal is the unification of the concepts of connectivity and similarity of nodes of a graph in a single real-valued distance matrix, which can be next used to create two-dimensional embeddings reflecting the above-mentioned similarities, by using existing multidimensional projection techniques.

We have presented, and explored, several ways of achieving the above goal, in terms of (1) encoding the connectivity information of the underlying network by modified adjacency matrices or shortest-path distances; (2) encoding node attributes by classical distance matrices based on multidimensional distance metrics; and (3) combining the

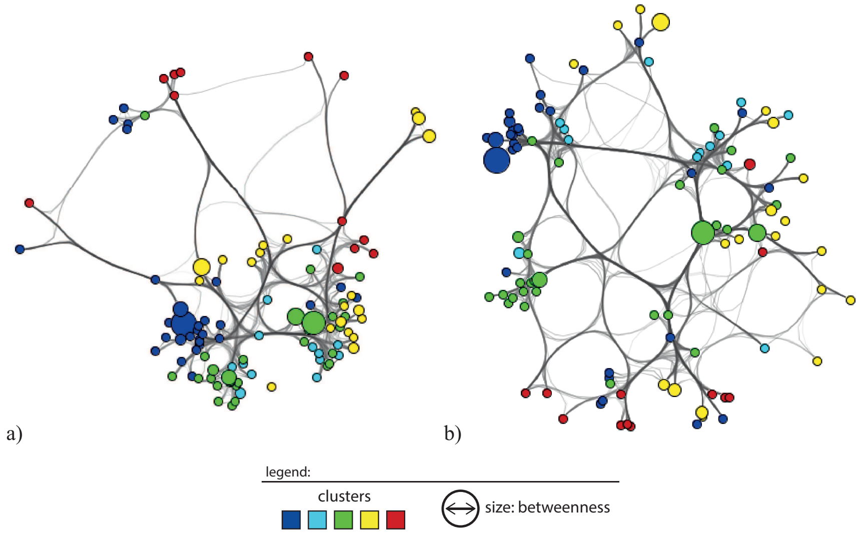


Figure 7.20: Alternative network visualization with edge bundling [182], using the same node coordinates shown in Fig. 7.6. Bundle opacities encode the edge counts in the respective bundles.

above-mentioned distance matrices in order to generate graph visualizations that reflect both the connectivity of their nodes as the similarity of nodes in terms of their attributes. These methods achieve different mixes of contributions of the relational information (edges) and attribute information (node and edge values) in the construction of the resulting 2D node-link layouts of the network.

One particularly interesting result regards the comparison of our graph visualization methods with classical force-based layout methods, which are, to date, among the most popular techniques for creating 2D embeddings of graphs. We have shown that the use of our proposed projection methods to create a preliminary 2D embedding of nodes, based on node connectivity and/or attribute similarity, can act as an effective preconditioner for force-based methods: By using such a preconditioner, the results of force-based embedders show a higher quality, measured both in terms of objective graph-layout quality metrics and in terms of qualitative insights concerning the resulting graph drawings.

Finally, we have shown how we can adapt the quality metrics proposed earlier (Sec. 7.3.5) for quantification of neighborhood preservation in multidimensional projections to handle multivariate graphs. This shows that it is possible to analyse the quality of drawings of such graphs at a spatially detailed level, and in ways which are identical to analysing the quality of general multidimensional projections. We believe that the unification of quality analysis of visualizations of graphs and multidimensional datasets is an interesting result, which, if refined further, can lead to important and useful results in the direction of exploration of complex hybrid multidimensional datasets.

## Chapter 8

# Discussions and Conclusion

We conclude our presentation of techniques for the visual explanation of multidimensional projections with an analysis of the key results presented in this thesis, along with their main added value and respective limitations. Additionally, we outline directions for future research and applications that are enabled by the work presented in this thesis.

## 8.1 Analysis of the Research Questions

As described in Chapter 1, we started this thesis with the hypothesis that multidimensional projections are effective tools for the exploration of high-dimensional datasets, but their usefulness does not reach its full potential due to the absence of explanatory techniques. For clarity, the general research question (originally presented in Section 1.2) is reproduced below:

> *How can we increase the added value of multidimensional projections of high-dimensional datasets with explanatory techniques that enable a wide range of users to interpret the information present in a projection in more effective ways?*

The general answer for this question – which has been proposed, described and developed throughout the entire body of this thesis – is that the visual representation of multidimensional projections should be improved in two different but complementary ways:

1. To make it clearer for the user where projection errors occur, what they mean and how severe they are, helping to evaluate whether a certain projection is indeed a good representation of the original dataset for the given task; and
2. To tell the user not only that certain points are similar (or not) by positioning them close to (or far away from) each other, but also why they are similar or not, using clear and descriptive visual explanations that can be used to reason about the original attributes of the dataset.

These two improvement areas are complementary – having one without the other leads to an incomplete solution. If we show and explain projection errors but do not explain the meaning of point positions in terms of the original data attributes, one will be able to 'filter out' wrongly projected points from a subsequent analysis, but not be able to understand the meaning of patterns created by the correctly projected points. Conversely, if we explain the meaning of projected points in terms of the high-dimensional variables but do not show local projection errors, one can be misled into using the visual explanation to reason about false patterns, which are due to projection errors rather than to actual patterns present in the high-dimensional data.

The above analysis also indicates the *order* in which error explanation and projection explanation techniques should be (jointly) used in a real-world visual analytics scenario. Given a multidimensional dataset, one would first use a multidimensional projection to generate a two-dimensional or three-dimensional projection of the respective data. Next, the error exploration techniques described in Chapters 3 and 4 should be used to understand the extend and distribution of projection errors. If these errors are too large overall and/or distributed over significant areas of the projection, then one should arguably not use the projection further, since it will very likely lead to too many wrong insights. If the errors affect a limited and/or localized projection area, then visual analysis of the remainder of the projection can be done. Next, visual explanatory techniques for two-dimensional projections (Chapter 6) and three-dimensional projections (Chapter 5) can be used to understand the meaning of the correctly-projected areas by explaining them in terms of the original high-dimensional attributes. The above workflow, or pipeline, can naturally be repeated by *e.g.* using different projection techniques and/or projection parameters to improve the projection quality in specific areas of interest, remove uninteresting observations and/or observations creating high projection errors, or simply focus on specific areas of interest, until the desired insights are obtained. This workflow follows the traditional visualization design mantra of overview and filtering first (with the detection and investigation error areas) and then details-on-demand (with the visual analysis of the remaining areas of interest in terms of their original attributes).

Since we could not, obviously, explore to exhaustion all the possible ways in which a projection can be improved, we instead focused on showing that it is indeed possible to improve the understanding of projections, and to provide insights that were not present before, by focusing on the two previously highlighted improvement areas. By applying the proposed techniques to various case studies with different datasets from different sources and domains, we have also shown that the improvements are not domain- or application-dependent, but can be observed in different situations, which points to the possibility of the generalization and wide-applicability of the results.

## 8.2 Design Decisions

Throughout our work, we have consistently followed a number of constraints and design decisions. As we believe that these design decisions support several important desirable features of explanatory visualizations of multidimensional data, we discuss them next.

**Projections as black-boxes.** One of our first choices of research method and design was to consider projection methods as black-boxes. We did not make any assumptions on specific constraints of the projection techniques, such as assuming that such techniques are of the multidimensional scaling variant (accepting distance matrices as inputs) or of the projection proper variant (accepting high-dimensional coordinates as inputs). We also did not assume any knowledge of the projection internals, *e.g.*, the fact that a projection is linear or not, or the fact it uses a certain neighborhood size. For us, a projection is simply a function mapping a set of high-dimensional observations to two-dimensional or three-dimensional points. The key advantage hereof is that all our results are directly applicable to *any* projection technique out there, without any modification, and in a technically straightforward manner. A second, equally important, advantage is that this approach focuses on the needs of a typical user who wishes or needs to use a projection to analyse a dataset of her own specific domain, but does not usually comprehend how a projection method works internally. In other words, our key goal is to explain the *result* of a projection, and not the projection *technique*.

However, this approach also comes with a limitation. Projection techniques get increasingly specialized beyond the original goal of globally and uniformly preserving distances and/or neighborhoods. For instance, projection techniques may choose to embed points so that distances between certain given observation-pairs are best preserved at the expense of other observation-pairs; or that certain distance ranges are better preserved than others; or to create as salient as possible separations between groups of related observations. For such projections, our techniques will likely indicate higher errors in certain areas, as compared to generic projections which aim to minimize errors uniformly for all observations. However, such errors may actually not be harmful, but actually instrumental, to using the projection for very specific tasks, such as the localization of strongly-related groups of observations. As such, the interpretation of errors seen in the projections should be always done by having a *task* in mind; and errors should be deemed negative only when their presence clearly affects the task at hand.

**Easy-to-understand error metrics.** In our study of projection errors, initally presented in Chapters 3 and 4, we introduced a number of relatively simple metrics for quantifying these errors, and visual encodings thereof, such as the false positive neighbors, false negative neighbors, group-related metrics, set-difference view, and sequence-difference views. These metrics are effective in explaining local errors in a projection in terms of both locality in the original high-dimensional space and the projection space. Obviously, many other types of errors can be imagined for a projection, and many other corresponding metrics can be designed to quantify these. However, the more complex such metrics become, the harder will be their visual interpretation and analysis. As projections are *already* very complex and abstract techniques, we believe that one should not make their explanations, which aim to lower their interpretation burden, too complex – otherwise they would be defeating their purpose. Additionally, more complex metrics serve more specialized scenarios, and our focus has been to cover the baseline explanatory scenarios first, which arguably are of interest for the widest category of users and their use-cases.

**Visual and computational scalability.** As outlined several times in this thesis, one of the key attractive aspects of projection techniques is their visual and computational scalability. Indeed, projections can handle datasets which are large in both observation and dimension count, with modern projection implementations scaling roughly linearly in computational effort with the size of the input dataset. Secondly, projections generate a structurally very simple representation – a two-dimensional or three-dimensional point cloud or scatterplot – which is fast to render and visually navigate, and also may accommodate hundreds of thousands of data points on a single screen. As such, our visual explanation tools were designed to keep the visual and computational scalability features of projections. This way, an integrated end-to-end visual analytics pipeline featuring joint projection and visual explanation tools will be visually and computationally scalable, and thus interesting for real-world usage. We achieved this scalability by a number of design decisions and techniques, as follows. First, we favored a simple visual representation consisting of (color coded) scatterplots, edge bundles, synthesized images, and legends. All these techniques are visually scalable, as they generate space-filling representations where virtually every pixel can be counted on to encode information. Secondly, these representations do a good job in reducing or minimizing visual clutter, which is also instrumental to increase visual scalability. Thirdly, we used a mix of highly computationally scalable techniques for the generation of our visualizations, such as efficient spatial search structures [6], GPU-based image synthesis techniques [28, 182], and GPU-accelerated edge bundling techniques [85]. All these techniques ensure that we can generate our visual explanations within fractions of a second for datasets having tens of thousands of observations and tens of dimensions on a typical desktop computer, which allows for interactive exploration.

## 8.3 Advantages and Limitations

To provide more insight into our level of answering the original research questions, we detail next the main advantages and limitations of our proposed techniques with respect to these questions.

### 8.3.1 Sub-question #1: Understanding projection errors

Our main goal here has been to provide generic, computationally scalable, and easy-to-understand visual metaphors that explain different types of errors that occur in multidimensional projections. To this end, we have provided ways to measure and depict false neighbors and missing neighbors caused by distance errors, at both individual point level and at the level of groups of points, in 2D projections (Chapter 3); ways to measure and depict the level of preservation of $k$-nearest neighborhoods at local point level in 2D projections (Chapter 3); and ways to measure and depict false and missing neighbors caused by distance errors at point level in 3D projections (Chapter 5). All these techniques are simple to implement and computationally scalable. For 2D projections, visual scalability is ensured by using image-based techniques to encode the resulting error maps as continuous 2D fields (space-filling techniques). This design also allows us to easily compute multiscale representations of our various error maps to focus on the most salient error patterns and/or to increase visual scalability and decrease clutter.

The ease of understanding of the proposed techniques is, arguably, harder to quantify. We have used the proposed techniques overall with a limited group of users (about 15 persons in total), but with a large number of representative datasets and corresponding questions (over 25 in total). The qualitative feedback collected from observing the users and discussing feedback with them was that the proposed error presentation techniques are quite easy to learn and understand, as they involve very simple and typically already known visual metaphors (color-coded scatterplots, legends, colored heatmaps, and edge bundles). The key difficulty in understanding was actually observed to be related to the concept of projection error, whose comprehension requires a certain familiarity with the process of dimensionality reduction. While researchers in machine learning, statistics, or data visualization had no problems at all to follow the error explanations, typical end-users coming from outside these fields would comprehend where an error is, which points this error affects, and how to compare different error levels, but not necessarily understand what to do next to reduce these errors or eliminate them from subsequent data interpretations.

Related to the ease of use of our techniques, we should mention that all of them involve a (small) amount of interactive exploration. This poses a corresponding (small) burden on the users. However, we noticed no instance when this exploration would be perceived as complex, hard to learn, or tedious to use. This is arguably related to the minimal nature of the proposed interactive tools – brushing, lasso and bounding-box selection, 3D trackball manipulation, and simple GUI-based preset choices for various visualization styles. All these operations are well known by a large spectrum of users, can be accomplished with minimal effort, and require limited precision during the interaction.

### 8.3.2 Sub-question #2: Understanding projections

Our main goal here has been to provide generic, computationally scalable, and easy-to-understand visual metaphors that explain (parts of) a multidimensional projection in terms of the original high-dimensional variables of the projected dataset. To this end, we have provided two different explanation designs.

For 2D projections, we used the same design elements employed to explain projection errors – that is, shaded cushions, color coding, image synthesis, and legends. In line with the comments provided above in Sec. 8.3.1, these techniques are generic, and visually and computationally scalable. One main novel point for the visual explanation of 2D projections was, in our view, the creation of *implicit* regions in a projection that explain subsets of points in terms of the similarity of one or a few variables. This leads to visualizations which strongly resemble the way humans annotate a map or diagram or other drawing consisting of a set of graphical symbols, by surrounding regions that allow a simple explanation, and writing the explanation atop of these regions. However, this technique also has its limitations. When the number of regions that require different explanations is too large, roughly over a dozen for a typical computer screen, our technique may not be sufficiently visually scalable. More importantly, our technique only covers explanation in terms of variable similarity. This naturally follows the fact that projections place similar observations close to each other, thus naturally give birth to precisely the type of regions we are creating in our visualizations. However, one can easily imagine many other ways in which one can 'divide' a 2D projection into parts that can be independently explained, *e.g.*, by explaining why a region is different from all surrounding regions. While our visual design accommodates such explanation types too, they are still open to be explored in the future.

For 3D projections, we extended the design elements promoted by Broeksema *et al.* [24, 23] for the global explanation of 2D projections (Chapter 5). By global explanation, we mean that our design only indicates how the original high-dimensional variables map to the entire projection space – in contrast to the local explanation design shown in Chapter 6 which divides the projection space into regions that are given different explanations. As such, while both designs presented in Chapters 5 and 6 are attribute-centric – in the sense that they explain a projection by the names and/or values of the original high-dimensional attributes, the explanation of 3D projections (Chapter 5) is global, while the explanation of 2D projections (Chapter 6) is local. The design choice to use a global explanation for 3D projections has been determined chiefly by the fact that local explanations would create too complex 3D patterns to be easily visualized. Indeed, such patterns would give rise to multiple 3D shapes (groups of points, point hulls, or similar) whose visual exploration in 3D would be, arguably, highly impeded by occlusion. To limit occlusion, we restricted our 3D explanations designs to metaphors specifically designed not to crowd and/or clutter the 3D scatterplot created by the projection – axis legends, screen-axis legends, and viewpoint legends. All such legends, with the exception of the axis legends, are placed outside the 3D scatterplot visualization, so do not create any occlusion. The axis legends are minimal in design – a set of one-dimensional curves embedded in 3D – and therefore create negligible additional clutter.

While our 3D explanatory visualizations demonstrably bring more insight than visualizing 'raw' 3D scatterplots, they also have several limitations. First and foremost, they are global, as said earlier. As such, if one locates, for example, a particular formation (outlier, cluster) of points in a specific region of the 3D projection space, explaining it in terms of variables and their values requires a certain amount of interactive projection exploration in terms of axis alignment and color-coding (for details, we refer to Chapter 5). This is in high contrast to our 2D projection explanations which require no user interaction (Chapter 6). Secondly, 3D projections are, by their very nature, encumbered by occlusion. Our explanatory techniques, while not *increasing* this clutter further on, do not also decrease it, and thus still suffer from its presence. This is, to our understanding to date, a required price to pay for the added precision of 3D projections.

## 8.4 Future Work

Given the huge scope, increasing abundance, and prominence in science and technology of multidimensional data and related analysis problems, and the well-known scalability and genericity of multidimensional projections as exploration tools for such data, there are many future possible extensions and applications of our work presented here on the design of visual exploratory and explanatory techniques for multidimensional data. Below we outline only the most salient ones.

**Data size and types:** Our work has mainly considered quantitative and relational datasets having up to tens of thousands of observations and tens of dimensions. Clearly, much larger datasets in terms of both observation count and dimension count exist, are easily to obtain, and are interesting to analyse for their involved stakeholders. Exploring how our visual explanatory tools can be adapted to cope with such massive datasets is a low-hanging fruit for further exploration. Additionally, we can easily consider the extension and/or adaption of our techniques to explain non-quantitative, non-relational datasets, such as datasets having categorical and ordinal attribute types. Projection techniques and explanatory techniques have been investigated for such datasets [24, 23]. Finally, our techniques have only handled static time-independent data. Exploring time-dependent data brings a sizeable set of non-trivial theoretic and practical challenges: Data sets become significantly larger; time requires a suitable mapping to a visual dimension; and defining, detecting, and explaining relevant spatio-temporal patterns and corresponding errors in multidimensional data is, by itself, a barely studied problem. Providing simple and effective visual explanations for multidimensional temporal data is an important current, and future, challenge to visual analytics.

**Improving projections:** Our work has mainly focused on *explaining* projections, in terms of errors and original high-dimensional variables. One of the use-cases related to providing this explanation is that, when errors and/or other patterns of interest are found by using our tools, users can next perform actions to focus on the interesting parts of the data space for further exploration, *e.g.* by removing the causes of the errors and redoing the projection, or by selecting specific subsets of observations and/or dimensions to further explore. So far, our explicit support for the entire round-trip visualization improvement and refinement pipeline is limited to the detection of problematic or interesting events, but not to their further elimination or examination. Many future work directions exist here. For instance, visual cues could be provided to indicate, in a projection, the actions one needs to take locally or globally to decrease errors in a specific area of interest; and next, interactive aids could be provided to execute these actions in the visual space, thereby reducing the round-trip execution time to a minimum.

**Evaluation:** As already discussed in Chapter 4, proposed solutions for analysing projection errors and explaining them regarding the original attributes of the data may work differently for different scenarios. To explore this variability, we have used, throughout our work, a relatively small but – we argue – comprehensive set of projections, datasets, and parameter settings. The main reason behind limiting the number of samples of this space of possibilities is computational complexity: For instance, evaluating our four different error types for ten projection techniques, each tested on ten different datasets, each using ten different parameter settings, would produce four thousand different projection images. Clearly, analysing and discussing such a number of examples is prohibitive. To manage this complexity, we chose therefore to limit ourselves to a smaller subset of well-known, robust, easy to use, and scalable projection techniques; consider a small subset of parameter values which are close to the optimal configurations for these techniques; and treat a limited number of datasets which are well-known in the literature, so that comparisons with existing work are easier to do. Separately, this limitation in variability of the studied projection techniques, parameter settings, and datasets makes it easier to relate and compare our results presented in different chapters.

However, it is entirely possible that unknown strengths and/or limitations of our proposed techniques were still not fully explored with the presented use-cases. Hence, we acknowledge the importance of performing, as future work, more strict and more thorough evaluations and validations. One example hereof is the evaluation of the proposed error metrics with artificial datasets specifically crafted to bring to light possible limitations, such as the already discussed sensitivity to outliers. It is also important to evaluate different combinations of projections, parameter settings and datasets, to validate the quality of the results presented in this thesis and to understand how they behave in the widest possible range of situations. The data-driven framework for evaluation of visual quality measures proposed by Sedlmair and Aupetit [162] is one interesting possibility for supporting future work in this direction.

**Integration and validation:** The last, but definitely not the least, important point for future work concerns the integration of our visual explanatory tools together with other data mining and analysis tools such as filters, statistical estimators, cluster and pattern mining algorithms, into fully-fledged end-to-end visual analytics systems. Such systems are, on the one hand, the prime means to test and validate the added value of our proposed techniques in the context of real-world applications, users, and use-cases. On the other hand, the design and deployment of such systems, and the generation of useful insights for their targeted end-users, is the ultimate goal that our work aims to support, and the core reason-to-be of the entire research in data visualization and visual analytics.

# Bibliography

[1] H. Abdi and D. Valentin. Multiple correspondence analysis. In: N. J. Salkind (ed.), *Encyclopedia of Measurement and Statistics*, pp. 651–657. Thousand Oaks (CA), 2007.

[2] J. Abello and F. van Ham. Matrix Zoom: A visual interface to semi-external graphs. In: *Proceedings of the IEEE Symposium on Information Visualization* (INFOVIS), pp. 183–190. IEEE, 2004.

[3] A.-C. Achilles and P. Ortyl. The Collection of Computer Science Bibliographies, 2013. *Available at:* `http://liinwww.ira.uka.de/bibliography/`.

[4] D. Archambault, T. Munzner, and D. Auber. GrouseFlocks: Steerable exploration of graph hierarchy space. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):900–913, 2008.

[5] A. Aris and B. Shneiderman. Network Visualization by Semantic Substrates. *Information Visualization*, 12(5):733–740, 2007.

[6] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.
*Available at:* `http://www.cs.umd.edu/ mount/ANN`.

[7] D. Auber. Tulip — a huge graph visualization framework. In: *Graph Drawing Software*, Mathematics and Visualization series, pp. 105–126. Springer Berlin Heidelberg, 2004.

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

[9] A. T. Azar and A. E. Hassanien. Dimensionality reduction of medical big data using neural-fuzzy classifier. *Soft computing*, 19(4):1115–1127, 2014.

[10] V. R. Basili, L. C. Briand, and W. L. Melo. A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on Software Engineering*, 22(10):751–761, 1996.

[11] H. Bauer and K. Pawelzik. Quantifying the neighborhood preservation of self-organizing feature maps. *IEEE Transactions on Neural Networks*, 3(4):570–579, 1992.

[12] R. Becker, W. Cleveland, and M. Shyu. The visual design and control of trellis display. *Journal of Computational and Graphical Statistics*, 5(2):123–155, 1996.

[13] M. W. Berry. *Survey of text mining: Clustering, classification, and retrieval*. Springer, 2004.

[14] E. Bertini, A. Tatu, and D. Keim. Quality metrics in high-dimensional data visualization: an overview and systematization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2203–2212, 2011.

[15] A. Bezerianos, F. Chevalier, P. Dragicevic, N. Elmqvist, and J.-D. Fekete. GraphDice: A System for Exploring Multivariate Social Networks. *Computer Graphics Forum*, 29(3):863–872, 2010.

[16] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. U. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424(4-5):175–308, 2006.

[17] I. Borg and P. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer, 2005.

[18] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy. *Polygon Mesh Processing*. A. K. Peters, 2010.

[19] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[20] U. Brandes and C. Pich. Eigensolver methods for progressive multidimensional scaling of large data. In: M. Kaufmann and D. Wagner (eds.), *Graph Drawing – 14th International Symposium (GD 2006)*, volume 4372 of *Lecture Notes in Computer Science*, pp. 42–53. Springer, 2007.

[21] M. Brehmer and T. Munzner. A multi-level typology of abstract visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2376–2385, 2013.

[22] M. Brehmer, M. Sedlmair, S. Ingram, and T. Munzner. Visualizing dimensionally-reduced data: Interviews with analysts and a characterization of task sequences. In: *Proceedings of the Workshop on Beyond Time and Errors: Novel Evaluation Methods for Visualization* (BELIV), pp. 1–8. ACM, 2014.

[23] B. Broeksema, T. Baudel, A. C. Telea, and P. Crisafulli. Decision exploration lab: A visual analytics solution for decision management. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):1972–1981, 2013.

[24] B. Broeksema, A. C. Telea, and T. Baudel. Visual analysis of multidimensional categorical data sets. *Computer Graphics Forum*, 32(8):158–169, 2013.

[25] M. Bronstein, A. Bronstein, R. Kimmel, and I. Yavneh. Multigrid multidimensional scaling. *Numerical Linear Algebra with Applications*, 13(2–3):149–171, 2006.

[26] H. Byelas and A. C. Telea. Visualizing metrics on areas of interest in software architecture diagrams. In: *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis)*, pp. 33–40. IEEE, 2009.

[27] H. Byelas and A. C. Telea. Visualizing multivariate attributes on software diagrams. In: *Proceedings of the European Conference on Software Maintenance and Reengineering* (CSMR), pp. 335–338. IEEE, 2009.

[28] T.-T. Cao, K. Tang, A. Mohamed, and T.-S. Tan. Parallel banding algorithm to compute exact distance transform with the GPU. In: *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (I3D), pp. 83–90. ACM, 2010.

[29] M. Chalmers. A linear iteration time layout algorithm for visualising high-dimensional data. In: *Proceedings of IEEE Visualization* (VIS), pp. 127–131. IEEE, 1996.

[30] J. Chambers, W. Cleveland, B. Kleiner, and P. Tukey. *Graphical Methods for Data Analysis*. Wadsworth, 1983.

[31] Y.-H. Chan, C. D. Correa, and K.-L. Ma. Regression cube: A technique for multidimensional visual exploration and interactive pattern finding. *ACM Transactions on Interactive Intelligent Systems*, 4(1):7, 2014.

[32] M.-C. Chang, F. Leymarie, and B. Kimia. Surface reconstruction from point clouds by transforming the medial scaffold. *Computer Vision and Image Understanding*, 113(11):1130–1146, 2009.

[33] S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994.

[34] J. Claessen and J. J. van Wijk. Flexible linked axes for multivariate data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2310–2316, 2011.

[35] D. B. Coimbra, R. M. Martins, T. T. A. T. Neves, A. C. Telea, and F. V. Paulovich. Explaining three-dimensional dimensionality reduction plots. *Information Visualization*, 2015. *Available at:* `http://dx.doi.org/10.1177/1473871615600010`.

[36] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[37] K. A. Cook and J. J. Thomas. Illuminating the path: The research and development agenda for visual analytics. Technical report, Pacific Northwest National Laboratory (PNNL), Richland, WA (US), 2005.

[38] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

[39] L. Costa and R. Cesar. *Shape Analysis and Classification*. CRC Press, 2001.

[40] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Chapman & Hall, 2001.

[41] A. M. Cuadros, F. V. Paulovich, R. Minghim, and G. P. Telles. Point Placement by Phylogenetic Trees and its Application to Visual Analysis of Document Collections. In: *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 99–106. IEEE, 2007.

[42] R. da Silva, P. Rauber, R. M. Martins, R. Minghim, and A. C. Telea. Attribute-based visual explanation of multidimensional projections. In: *Proceedings of the International EuroVis Workshop on Visual Analytics* (EuroVA), pp. 134–139. Eurographics Association, 2015.

[43] T. Dang, L. Wilkinson, and A. Anand. Stacking graphic elements to avoid over-plotting. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1044–1052, 2010.

[44] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[45] S. Diehl. *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Springer, 2008.

[46] S. Diehl and A. C. Telea. Multivariate networks in software engineering. In: A. Kerren, H. C. Purchase, and M. O. Ward (eds.), *Multivariate Network Visualization – Dagstuhl Seminar #13201, 2013*, volume 8380 of *Lecture Notes in Computer Science*, pp. 13–36. Springer, 2014.

[47] S. Ding, H. Zhu, W. Jia, and C. Su. A survey on feature extraction for pattern recognition. *Artificial Intelligence Review*, 37(3):169–180, 2012.

[48] P. A. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[49] G. Ellis and A. Dix. A taxonomy of clutter reduction for information visualisation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1216–1223, 2007.

[50] N. Elmqvist, P. Dragicevic, and J.-D. Fekete. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1141–1148, 2008.

[51] A. Endert, P. Fiaux, and C. North. Semantic interaction for visual text analytics. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI), pp. 473–482. ACM, 2012.

[52] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD), pp. 226–231. AAAI, 1996.

[53] R. Etemadpour, R. Motta, J. de Souza, R. Minghim, F. de Oliveira, M. Cristina, and L. Linsen. Perception-based evaluation of projection methods for multidimensional data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 21(1):81–94, 2015.

[54] S. Fabricant. *Spatial metaphors for browsing large data archives*. PhD thesis, University of Colorado Boulder, 2000.

[55] C. Faloutsos and K.-I. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *SIGMOD Record*, 24(2):163–174, 1995.

[56] S. G. Finlayson, P. LePendu, and N. H. Shah. Building the graph of medicine from millions of clinical narratives. *Scientific Data*, 1, 2014.

[57] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41, 1977.

[58] Freephoto. Free stock photo collection, 2013. *Available at:* `www.freephoto.com`. *Last access:* 15/07/2013.

[59] Y. Frishman and A. Tal. Multilevel graph layout on the GPU. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1310–1319, 2007.

[60] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[61] E. Gansner, Y. Hu, S. North, and C. Scheidegger. Multilevel agglomerative edge bundling for visualizing large graphs. In: *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis)*, pp. 187–194, 2011.

[62] E. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In: *Graph Drawing*, pp. 239–250. Springer, 2004.

[63] X. Geng, D.-C. Zhan, and Z.-H. Zhou. Supervised nonlinear dimensionality reduction for visualization and classification. *IEEE Transactions on Systems, Man and Cybernetics*, 35(6):1098–1107, 2005.

[64] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[65] GitHub. Online project hosting using Git, 2015.
*Available at:* `https://github.com/`.

[66] M. Gleicher. Explainers: Expert explorations with crafted projections. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2042–2051, 2013.

[67] M. Gletsos, S. G. Mougiakakou, G. K. Matsopoulos, K. S. Nikita, A. S. Nikita, and D. Kelekis. A computer-aided diagnostic system to characterize CT focal liver lesions: design and optimization of a neural network classifier. *IEEE Transactions on Information Technology in Biomedicine*, 7(3):153–162, 2003.

[68] P. A. Gloor, J. Krauss, S. Nann, K. Fischbach, and D. Schoder. Web Science 2.0: Identifying Trends through Semantic Social Network Analysis. In: *Proceedings of the 2009 International Conference on Computational Science and Engineering*, pp. 215–222, 2009.

[69] J. Gower, S. Lubbe, and N. Roux. *Understanding biplots*. Wiley, 2011.

[70] GraphViz. The GraphViz graph visualization software, 2014.
*Available at:* `http://www.graphviz.org/`.

[71] M. Greenacre. *Biplots in practice*. Fundacion BBVA, 2010.

[72] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[73] C. Hansen and C. J. Johnson. *The Visualization Handbook*. Elsevier, 2005.

[74] M. Harrower and C. A. Brewer. ColorBrewer.org: an online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003.

[75] J. Heer and D. Boyd. Vizster: Visualizing Online Social Networks. In: *Proceedings of the IEEE Symposium on Information Visualization* (INFOVIS), pp. 32–39. IEEE, 2005.

[76] N. Henry and J.-D. Fekete. MatrixExplorer: A Dual-Representation System to Explore Social Networks. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):677–684, 2006.

[77] N. Henry, J.-D. Fekete, and M. McGuffin. NodeTrix: A Hybrid Visualization of Social Networks. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1302–1309, 2007.

[78] I. Herman, G. Melançon, and M. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[79] N. Heulot, M. Aupetit, and J.-D. Fekete. Proxilens: Interactive exploration of high-dimensional data using projections. In: *Proceedings of the EuroVis Workshop on Visual Analytics using Multidimensional Projections*, pp. 11–15, 2013.

[80] G. E. Hinton and S. T. Roweis. Stochastic neighbor embedding. In: *Advances in neural information processing systems*, pp. 833–840, 2002.

[81] P. Hoffman, G. Grinstein, K. Marx, I. Grosse, and E. Stanley. DNA visual and analytic data mining. In: *Proceedings of the IEEE Visualization* (VIS), pp. 437–ff. IEEE, 1997.

[82] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[83] M. Huisman and M. A. J. Duijn. *Models and Methods in Social Network Analysis*, chapter: Software for Social Network Analysis, pp. 270–316. Cambridge University Press, 2005.

[84] C. Hurter, O. Ersoy, S. Fabrikant, T. Klein, and A. C. Telea. Bundled visualization of dynamic graph and trail data. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1141–1157, 2014.

[85] C. Hurter, O. Ersoy, and A. C. Telea. Graph bundling by kernel density estimation. *Computer Graphics Forum*, 31(3):865–874, 2012.

[86] C. Hurter, R. Taylor, S. Carpendal, and A. C. Telea. Color tunneling: Interactive exploration and selection in volumetric datasets. In: *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis)*, pp. 225–232, 2014.

[87] C. Hurter, B. Tissoires, and S. Conversy. FromDaDy: Spreading data across views to support iterative exploration of aircraft trajectories. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1017–1024, 2009.

[88] S. Ingram, T. Munzner, V. Irvine, M. Tory, S. Bergner, and T. Moller. DimStiller: Workflows for dimensional analysis and reduction. In: *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 3–10. IEEE, 2010.

[89] S. Ingram, T. Munzner, and M. Olano. Glimmer: Multilevel MDS on the GPU. *IEEE Transactions on Visualization and Computer Graphics*, 15(2):249–261, 2009.

[90] A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multidimensional geometry. In: *Proceedings of the IEEE Visualization* (VIS), pp. 361–378. IEEE Computer Society Press, 1990.

[91] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[92] A. K. Jain, M. N. Murthy, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[93] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In: *Proceedings of the ACM International Conference on Multimedia* (MM), pp. 675–678. ACM, 2014.

[94] S. Johansson and J. Johansson. Interactive dimensionality reduction through user-defined combinations of quality metrics. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):993–1000, 2009.

[95] P. Joia, F. V. Paulovich, D. Coimbra, J. A. Cuminato, and L. G. Nonato. Local affine multidimensional projection. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2563–2571, 2011.

[96] P. Joia, F. Petronetto, and L. G. Nonato. Uncovering representative groups in multidimensional projections. *Computer Graphics Forum*, 34(3):281–290, 2015.

[97] I. Jolliffe. *Principal Component Analysis*, p. 487. Series in Statistics series. Springer, 2002.

[98] F. Jourdan and G. Melançon. Multiscale hybrid MDS. In: *Proceedings of the IEEE Symposium on Information Visualization* (INFOVIS), pp. 388–393. IEEE, 2004.

[99] E. Kandogan. Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. In: *Proceedings of the IEEE Symposium on Information Visualization* (INFOVIS), pp. 9–12. IEEE, 2000.

[100] E. Kandogan. Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD), pp. 107–116. ACM, 2001.

[101] E. Kandogan. Just-in-time annotation of clusters, outliers, and trends in point-based data visualizations. In: *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 73–82. IEEE, 2012.

[102] D. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann (eds.). *Mastering the information age: Solving problems with visual analytics (VisMaster)*. Eurographics Association, 2010.

[103] Y. Koren, L. Carmel, and D. Harel. ACe: A fast multiscale eigenvector computation for drawing huge graphs. In: *Proceedings of the IEEE Symposium on Information Visualization* (INFOVIS), pp. 137–144. IEEE, 2002.

[104] Y. Koren, L. Carmel, and D. Harel. Drawing huge graphs by algebraic multigrid optimization. *Multiscale Modeling & Simulation*, 1(4):645–673, 2003.

[105] J. Krause, A. Perer, and E. Bertini. INFUSE: interactive feature selection for predictive modeling of high dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1614–1623, 2014.

[106] J. B. Kruskal and J. M. Landwehr. Icicle plots: Better displays for hierarchical clustering. *The American Statistician*, 37(2):162–168, 1983.

[107] M. Lanza and R. Marinescu. *Object-Oriented Metrics in Practice*. Springer, 2006.

[108] F. Lederman. Parvis – parallel coordinates visualisation, 2012.
*Available at:* `http://www.mediavirus.org/parvis/`.

[109] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry. Task taxonomy for graph visualization. In: *Proceedings of the Workshop on Beyond Time and Errors: Novel Evaluation Methods for Visualization* (BELIV), pp. 1–5. ACM, 2006.

[110] D. J. Lehmann, G. Albuquerque, M. Eisemann, M. Magnor, and H. Theisel. Selecting coherent and relevant plots in large scatterplot matrices. *Computer Graphics Forum*, 31(6):1895–1908, 2012.

[111] S. Lespinats and M. Aupetit. CheckViz: Sanity check and topological clues for linear and non-linear mappings. *Computer Graphics Forum*, 30(1):113–125, 2011.

[112] M. Levandowsky and D. Winter. Distance between sets. *Nature*, 234(5323):34–35, 1971.

[113] J. M. Lewis, L. Van Der Maaten, and V. R. de Sa. A behavioral investigation of dimensionality reduction. In: *Proceedings of the Annual Conference of the Cognitive Science Society* (CogSci), pp. 671–676. Cognitive Science Society, 2012.

[114] C.-T. Li and S.-D. Lin. Egocentric Information Abstraction for Heterogeneous Social Networks. In: *Proceedings of the International Conference on Advances in Social Network Analysis and Mining* (ASONAM), pp. 255–260. IEEE, 2009.

[115] J. Li and J. Z. Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1075–1088, 2003.

[116] M. Lichman. UCI machine learning repository, 2013.
*Available at:* `http://archive.ics.uci.edu/ml`.

[117] R. M. Martins, G. F. Andery, H. Heberle, F. V. Paulovich, A. de Andrade Lopes, H. Pedrini, and R. Minghim. Multidimensional projections for visual analysis of social networks. *Journal of Computer Science and Technology*, 27(4):791–810, 2012.

[118] R. M. Martins, D. Coimbra, R. Minghim, and A. C. Telea. Visual analysis of dimensionality reduction quality for parameterized projections. *Computers & Graphics*, 41:26–42, 2014.

[119] R. M. Martins, R. Minghim, and A. C. Telea. Explaining Neighborhood Preservation for Multidimensional Projections. In: *Computer Graphics and Visual Computing (CGVC)*. The Eurographics Association, 2015.
*Best Student Paper Award*.

[120] A. Mead. Review of the development of multidimensional scaling methods. *The Statistician*, 41(1):27–39, 1992.

[121] P. Meirelles, C. Santos, J. Miranda, F. Kon, A. Terceiro, and C. Chavez. A study of the relationships between source code metrics and attractiveness in free software projects. In: *Proceedings of the Brazilian Symposium on Software Engineering* (SBES), pp. 11–20. IEEE, 2010.
*Data available at:* `http://ccsl.ime.usp.br/mangue/data`. *Last access:* 15/07/13.

[122] R. Minghim, F. V. Paulovich, and A. A. Lopes. Content-Based Text Mapping using Multi-Dimensional Projections for Exploration of Document Collections. In: *SPIE Proceedings: Visualization and Data Analysis*, volume 6060, pp. S1–S12, 2006.

[123] R. Motta, A. Lopes, B. Nogueira, S. Rezende, M. Jorge, and M. de Oliveira. Comparing relational and non-relational algorithms for clustering propositional data. In: *Proceedings of the ACM Symposium on Applied Computing*, pp. 150–155. ACM, 2013.

[124] R. Motta, R. Minghim, A. Lopes, and M. Oliveira. Graph-based measures to assist user assessment of multidimensional projections. *Neurocomputing*, 150:583–598, 2015.

[125] T. Munzner. *Visualization Analysis and Design*. CRC Press, 2015.

[126] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1(2012):28, 2008.

[127] G. M. Namata, B. Staats, L. Getoor, and B. Shneiderman. A Dual-View Approach to Interactive Network Visualization. In: *Proceedings of the ACM Conference on Information and Knowledge Management*, pp. 939–942. ACM, 2007.

[128] G. B. Newby. Empirical study of a 3D visualization for information retrieval tasks. *Journal of Intelligence Information Systems*, 18(1):31–53, 2002.

[129] A. Nocaj and U. Brandes. Computing Voronoi treemaps: Faster, simpler, and resolution-independent. *Computer Graphics Forum*, 31(3):855–864, 2012.

[130] A. Nocaj and U. Brandes. Organizing search results with a reference map. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2546–2555, 2012.

[131] M. Norman and D. Whalen. IEEE Vis'08 contest data, 2013.
*Available at:* `http://sciviscontest.ieeevis.org/2008`.

[132] S. Oeltze, H. Doleisch, H. Hauser, P. Muigg, and B. Preim. Interactive visual analysis of perfusion data. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1392–1399, 2007.

[133] O. N. Oliveira Jr., T. T. A. T. Neves, F. V. Paulovich, and M. C. F. de Oliveira. Where chemical sensors May assist in clinical diagnosis exploring 'big data'. *Chemistry Letters*, 43(11):1672–1679, 2014.

[134] K. Olsen, R. Korfhage, K. Sochats, M. Spring, and J. Williams. Visualization of a document collection: The VIBE system. *Information Processing & Management*, 29(1):69–81, 1993.

[135] P. Pagliosa, F. V. Paulovich, R. Minghim, H. Levkowitz, and L. G. Nonato. Projection inspector: Assessment and synthesis of multidimensional projections. *Neurocomputing*, 150:599–610, 2015.

[136] J. Paiva, W. Schwartz, H. Pedrini, and R. Minghim. Semi-supervised dimensionality reduction based on partial least squares for visual analysis of high dimensional data. *Computer Graphics Forum*, 31(3):1345–1354, 2012.

[137] T. Pang-Ning, M. Steinbach, and V. Kumar. *Introduction to data mining*. Pearson, 2006.

[138] F. V. Paulovich, D. Eler, J. Poco, C. Botha, R. Minghim, and L. G. Nonato. Piecewise Laplacian-based projection for interactive data exploration and organization. *Computer Graphics Forum*, 30(3):1091–1100, 2011.

[139] F. V. Paulovich, L. G. Nonato, R. Minghim, and H. Levkowitz. Least square projection: a fast high-precision multidimensional projection technique. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):564–575, 2008.

[140] F. V. Paulovich, M. C. F. Oliveira, and R. Minghim. The Projection Explorer: A Flexible Tool for Projection-based Multidimensional Visualization. In: *Proceedings of the Brazilian Symposium on Computer Graphics and Image Processing* (SIBGRAPI), pp. 27–36, 2007.

[141] F. V. Paulovich, C. T. Silva, and L. G. Nonato. Two-phase mapping for projecting massive data sets. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1281–1290, 2010.

[142] F. V. Paulovich, C. T. Silva, and L. G. Nonato. User-centered multidimensional projection techniques. *Computing in Science & Engineering*, 14(4):74–81, 2012.

[143] F. V. Paulovich, F. Toledo, G. P. Telles, R. Minghim, and L. G. Nonato. Semantic wordification of document collections. *Computer Graphics Forum*, 31(3):1145–1153, 2012.

[144] E. Pekalska, D. de Ridder, R. Duin, and M. Kraaijveld. A new method of generalizing Sammon mapping with application to algorithm speed-up. In: *Proceedings of the $5^{th}$ Annual Conference of the Advanced School for Computing and Imaging (ASCI)*, pp. 221–228, 1999.

[145] A. Perer and B. Shneiderman. Balancing Systematic and Flexible Exploration of Social Networks. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):693–700, 2006.

[146] H. Piringer, R. Kosara, and H. Hauser. Interactive f+c visualization with linked 2D/3D scatterplots. In: *Proceedings of the International Conference on Coordinated and Multiple Views in Exploratory Visulization* (CMV), pp. 49–60. IEEE, 2004.

[147] J. C. Platt. FastMap, MetricMap, and Landmark MDS are all Nyström algorithms. In: *Proceedings of the $10^{th}$ International Workshop on Artificial Intelligence and Statistics*, pp. 261–268. Society for Artificial Intelligence and Statistics, 2005.

[148] J. Poco, R. Etemadpour, F. V. Paulovich, T. Long, P. Rosenthal, M. Oliveira, L. Linsen, and R. Minghim. A framework for exploring multidimensional data with 3D projections. *Computer Graphics Forum*, 30(3):1111–1120, 2011.

[149] G. Pölzlbauer. Survey and comparison of quality measures for self-organizing maps. In: *Proceedings of the 5$^{th}$ Workshop on Data Analysis* (WDA), pp. 67–82, 2004.

[150] A. J. Pretorius and J. J. V. Wijk. Visual Analysis of Multivariate State Transition Graphs. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):685–692, 2006.

[151] R. Rao and S. K. Card. The Table Lens: Merging graphical and symbolic representations in an interactive focus+context visualization for tabular information. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI), pp. 318–322. ACM, 1994.

[152] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[153] M. Rumpf and A. C. Telea. A continuous skeletonization method based on level sets. In: *Proceedings of the Symposium on Data Visualisation* (VISSYM), pp. 151–ff. Eurographics Association, 2002.

[154] G. Salton. *Introduction to modern information retrieval*. McGraw, 1986.

[155] J. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, 1969.

[156] H. Sanftmann. *3D visualization of multivariate data*. PhD thesis, University of Stuttgart, Germany, 2014.
*Available at:* `http://elib.uni-stuttgart.de/opus/volltexte/2012/7807`.

[157] H. Sanftmann and D. Weiskopf. Illuminated 3D scatterplots. *Computer Graphics Forum*, 28(3):642–651, 2009.

[158] H. Sanftmann and D. Weiskopf. 3D scatterplot navigation. *IEEE Transactions on Visualization and Computer Graphics*, 18(11):1969–1978, 2012.

[159] T. Schreck, T. von Landesberger, and S. Bremm. Techniques for precision-based visual analysis of projected data. *Information Visualization*, 9(3):181–193, 2010.

[160] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3-D Graphics, $4^{rd}$ edition*. Kitware, Inc., 2006.

[161] H. J. Schulz, T. Nocke, M. Heitzler, and H. Schumann. A design space of visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2366–2375, 2013.

[162] M. Sedlmair and M. Aupetit. Data-driven evaluation of visual quality measures. *Computer Graphics Forum*, 34(3):201–210, 2015.

[163] M. Sedlmair, T. Munzner, and M. Tory. Empirical guidance on scatterplot and dimension reduction technique choices. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2634–2643, 2013.

[164] M. Sedlmair, A. Tatu, T. Munzner, and M. Tory. A taxonomy of visual cluster separation factors. *Computer Graphics Forum*, 31(3):1335–1344, 2012.

[165] J. A. Sethian. *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1996.

[166] Z. Shen, K.-L. Ma, and T. Eliassi-Rad. Visual analysis of large heterogeneous social networks by semantic and structural abstraction. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1427–1439, 2006.

[167] J. R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry*, 22(1–3):21–74, 2002.

[168] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In: *Proceedings of the IEEE Symposium on Visual Languages*, pp. 336–343. IEEE, 1996.

[169] V. Silva and J. Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Stanford University, 2004.

[170] V. Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In: *Advances in Neural Information Processing Systems*, volume 15, pp. 705–712. MIT Press, 2003.

[171] M. Smith, C. Giraud-Carrier, and N. Purser. Implicit Affinity Networks and Social Capital. *Information Technology and Management*, 10(2-3):123–134, 2009.

[172] C. Sorzano, J. Vargas, and A. Pascual-Montano. A survey of dimensionality reduction techniques, 2014. *Available at:* `http://arxiv.org/pdf/1403.2877`.

[173] R. O. Stehling, M. A. Nascimento, and A. Falcão. A compact and efficient image retrieval approach based on border/interior pixel classification. In: *Proceedings of the 11$^{th}$ International Conference on Information and Knowledge Management* (CIKM), pp. 102–109. ACM, 2002.

[174] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In: *Proceedings of the KDD Workshop on Text Mining*, volume 400, pp. 525–526. Boston, 2000.

[175] L. Tan, Y. Song, S. Liu, and L. Xie. ImageHive: Interactive content-aware image summarization. *IEEE Computer Graphics and Applications*, 32(1):46–55, 2012.

[176] A. Tatu, G. Albuquerque, M. Eisemann, P. Bak, H. Theisel, M. Magnor, and D. A. Keim. Automated analytical methods to support visual exploration of high-dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):584–597, 2010.

[177] A. Tatu, F. Maas, I. Farber, E. Bertini, T. Schreck, T. Seidl, and D. Keim. Subspace search and visualization to make sense of alternative clusterings in high-dimensional data. In: *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 63–72. IEEE, 2012.

[178] M. Tavanti and M. Lind. 2D vs 3D, implications on spatial memory. In: *Proceedings of the IEEE Symposium on Information Visualization* (INFOVIS), pp. 139–145. IEEE, 2001.

[179] E. Tejada, R. Minghim, and L. G. Nonato. On improved projection techniques to support visual exploration of multidimensional data sets. *Information Visualization*, 2(4):218–231, 2003.

[180] A. C. Telea. Combining extended table lens and treemap techniques for visualizing tabular data. In: *Proceedings of the Joint Eurographics / IEEE VGTC Conference on Visualization* (EuroVis), pp. 51–58. Eurographics Association, 2006.

[181] A. C. Telea. *Data visualization: principles and practice*. CRC Press, 2014. $2^{nd}$ edition.

[182] A. C. Telea and O. Ersoy. Image-based edge bundles: simplified visualization of large graphs. *Computer Graphics Forum*, 29(3):843–852, 2010.

[183] A. C. Telea and L. Voinea. An interactive reverse engineering environment for large-scale C++ code. In: *Proceedings of the ACM Symposium on Software Visualization* (SoftVis), pp. 67–76. ACM, 2008.

[184] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[185] A. Terceiro, J. Costa, J. Miranda, P. Meirelles, L. R. Rios, L. Almeida, C. Chavez, and F. Kon. Analizo: an extensible multi-language source code analysis and visualization toolkit. In: *Brazilian Conference on Software: Theory and Practice (Tools Session)*, 2010.

[186] W. S. Torgeson. Multidimensional scaling of similarity. *Psychometrika*, 30(4):379–393, 1965.

[187] E. R. Tufte. *Envisioning Information*. Graphics Press, 1990.

[188] J. W. Tukey and P. A. Tukey. Computer graphics and exploratory data analysis: An introduction. *The Collected Works of John W. Tukey: Graphics: 1965-1985*, 5:419, 1988.

[189] C. Turkay, P. Filzmoser, and H. Hauser. Brushing dimensions-a dual visual analysis model for high-dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2591–2599, 2011.

[190] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2008.

[191] University of Maryland. US counties dataset, 2014.
*Available at:* `http://archive.ics.uci.edu/ml`.

[192] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11):2431–2456, 2008.

[193] L. van der Maaten and G. Hinton. Visualizing non-metric similarities in multiple maps. *Machine Learning*, 87(1):33–35, 2012.

[194] L. van der Maaten, E. Postma, and H. van den Herik. Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, 10(1):66–71, 2009. Extended version available online: *www.iai.uni-bonn.de/∼jz/dimensionality_reduction_a_comparative_review.pdf*.

[195] R. van Liere and W. de Leeuw. Graphsplatting: Visualizing graphs as continuous fields. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):206–212, 2003.

[196] J. J. Van Wijk and H. van de Wetering. Cushion treemaps: Visualization of hierarchical information. In: *Proceedings of the IEEE Symposium on Information Visualization* (INFOVIS), pp. 73–82. IEEE, 1999.

[197] P. Velardi, R. Navigli, A. Cucchiarelli, and F. D'Antonio. A New Content-Based Model for Social Network Analysis. In: *Proceedings of the IEEE International Conference on Semantic Computing*, pp. 18–25. IEEE, 2008.

[198] J. Venna and S. Kaski. Neighborhood preservation in nonlinear projection methods: An experimental study. In: *Artificial Neural Networks/ICANN*, pp. 485–491. Springer, 2001.

[199] T. Villmann, R. Der, M. Herrmann, and T. M. Martinetz. Topology preservation in self-organizing feature maps: exact definition and measurement. *IEEE Transactions on Neural Networks*, 8(2):256–266, 1997.

[200] L. Voinea and A. C. Telea. Multiscale and multivariate visualizations of software evolution. In: *Proceedings of the ACM Symposium on Software Visualization* (SoftVis), pp. 115–124. ACM, 2006.

[201] L. Voinea, A. C. Telea, and J. J. van Wijk. CVSscan: Visualization of code evolution. In: *Proceedings of the ACM Symposium on Software Visualization* (SoftVis), pp. 47–56. ACM, 2005.

[202] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. van Wijk, J.-D. Fekete, and D. Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011.

[203] J. T.-L. Wang, X. Wang, K.-I. Lin, D. Shasha, B. A. Shapiro, and K. Zhang. Evaluating a class of distance-mapping algorithms for data mining and clustering. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD), pp. 307–311. ACM, 1999.

[204] M. Wattenberg. Visual Exploration of Multivariate Graphs. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI), pp. 811–819. ACM, 2006.

[205] S. Westerman, J. Collins, and T. Cribbin. Browsing a document collection represented in two-and three-dimensional virtual information space. *International Journal of Human-Computer Studies*, 62(6):713–736, 2005.

[206] S. Westerman and T. Cribbin. Mapping semantic information in virtual space: dimensions, variance and individual differences. *International Journal of Human-Computer Studies*, 53(5):765–787, 2000.

[207] L. Wilkinson, A. Anand, and R. Grossman. Graph-theoretic scagnostics. In: *Proceedings of the IEEE Symposium on Information Visualization* (INFOVIS), p. 21. IEEE, 2005.

[208] M. Williams and T. Munzner. Steerable, progressive multidimensional scaling. In: *Proceedings of the IEEE Symposium on Information Visualization* (INFOVIS), pp. 57–64. IEEE, 2004.

[209] P. C. Wong and J. Thomas. Visual analytics. *IEEE Computer Graphics and Applications*, 24(5):20–21, 2004.

[210] J. S. Yi, R. Melton, J. Stasko, and J. A. Jacko. Dust & magnet: multivariate information visualization using a magnet metaphor. *Information Visualization*, 4(4):239–256, 2005.

[211] X. Yuan, D. Ren, Z. Wang, and C. Guo. Dimension projection matrix/tree: Interactive subspace visual exploration and analysis of high dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2625–2633, 2013.

# List of Publications

The following publications resulted from the work presented in this thesis:

- R. M. Martins, D. Coimbra, R. Minghim, and A. C. Telea. Visual analysis of dimensionality reduction quality for parameterized projections. *Computers & Graphics*, 41:26–42, 2014

- R. M. Martins, R. Minghim, and A. C. Telea. Explaining Neighborhood Preservation for Multidimensional Projections. In: *Computer Graphics and Visual Computing (CGVC)*. The Eurographics Association, 2015.
  *Best Student Paper Award*

- D. B. Coimbra, R. M. Martins, T. T. A. T. Neves, A. C. Telea, and F. V. Paulovich. Explaining three-dimensional dimensionality reduction plots. *Information Visualization*, 2015. *Available at:* `http://dx.doi.org/10.1177/1473871615600010`

- R. da Silva, P. Rauber, R. M. Martins, R. Minghim, and A. C. Telea. Attribute-based visual explanation of multidimensional projections. In: *Proceedings of the International EuroVis Workshop on Visual Analytics* (EuroVA), pp. 134–139. Eurographics Association, 2015

- R. M. Martins, G. F. Andery, H. Heberle, F. V. Paulovich, A. de Andrade Lopes, H. Pedrini, and R. Minghim. Multidimensional projections for visual analysis of social networks. *Journal of Computer Science and Technology*, 27(4):791–810, 2012

# Acknowledgements

**Para minha mãe:** Você sempre colocou minha educação em primeiro lugar e, com isso, me abençoou para a vida inteira. Eu devo tudo a você. *Muito obrigado!*

**To my beautiful and beloved wife, Priscila:** You are the very definition of strength. Even if we stayed together for a thousand years, I would never stop learning the secrets of life from you. For always being there – *Thank you!*

**To Alex:** You have been a huge example of professional to me. Meeting you and working with you has been a turning point in my career. For all that you have taught me and for trusting me as your student – *Thank you!*

**To Rosane:** For all your support and guidance when things didn't seem to move anywhere, and for trusting me at hard times when even I didn't trust myself – *Thank you!*

**To Maldonado:** All of this started with you, and I would not want it any other way. It has always been an honor working with you, and I hope this honor goes on for a very long time still. *Thank you!*

**To Carol:** I love you! For being the best sister in the world – *Thank you!*

**To Manoel:** For designing the cover and being an awesome cousin – *Thank you!*

**To Danilo:** For being such an amazing friend and partner – *Thank you!*

**To Enrico:** Anyone who has ever bumped into you has been a lucky person, but to be your friend is indeed a blessing that happens very rarely in one's life. You are very important to us (me and Priscila). *Thank you!*

**To all my Groningen friends:** You showed me that life around the world is not that different from home, and that language and culture are only details when you have true friends! *Thank you!*

**To all group members from SVCG and VICG:** Working with all of you has been a great learning experience. For all the laughs, the lunches, the coffee breaks and the white-board discussions – *Thank you!*

**To my assessment committee:** For taking your time to read my thesis and give me such great improvement suggestions – *Thank you!*

My PhD was funded mainly by the São Paulo Research Foundation (FAPESP) – Process 2012/07722-9, with additional support from the National Council for Scientific and Technological Development (CNPq) – Processes 140125/2011-6 and 141700/2012-2, the Brazilian Coordination for the Improvement of Higher Education Personnel (CAPES) and NUFFIC, the Netherlands organisation for international cooperation in higher education – Process BEX 18959/12-1.

Rafael Messias Martins