MULTIDIMENSIONAL PROJECTIONS FOR THE VISUAL EXPLORATION OF MULTIMEDIA DATA

DANILO BARBOSA COIMBRA

.

Cover: Sport video visualization for the soccer match between Brazil against Netherlands in 2014 World Cup

Multidimensional Projections for the Visual Exploration of Multimedia Data

Danilo Barbosa Coimbra
Thesis University of Groningen

isbn 978-90-367-8650-8 (printed version)
isbn 978-90-367-8649-2 (electronic version)

university of
groningen

Multidimensional Projections for the Visual Exploration of Multimedia Data

PhD thesis

to obtain the degree of PhD at the
University of Groningen
on the authority of the
Rector Magnificus Prof. E. Sterken
and in accordance with
the decision by the College of Deans.

This thesis will be defended in public on

Friday 18 March 2016 at 16.15 hours

by

Danilo Barbosa Coimbra

born on 11 June 1985
in Ribeirão Preto, Brazil

THESIS SUMMARY

The continuously advent of new technologies have made a rich and growing type of information sources available to analyses and investigation. In this context, multidimensional data analysis is considerably important when dealing with such large and complex datasets. Among the possibilities when analyzing such kind of data, applying visualization techniques can help the user find and understand patters, trends and establish new goals. Some applications examples of visualization of multidimensional data analysis goes from image classification, semantic word clouds, cluster analysis of document collection to exploration of multimedia content.

This thesis presents several visualization methods to interactively explore multidimensional datasets aimed from specialized to casual users, by making use of both static and dynamic representations created by multidimensional projections. Firstly, we present multidimensional projection technique which faithfully preserves distance and can handle any type of high-dimensional data, demonstrating applications scenarios in both multimedia and text documents collections. Next, we address the task of interpreting projections in 2D, by calculating neighborhood errors. Hereafter, we present a set of interactive visualizations that aim to help users with these tasks by revealing the quality of a projection in 3D, applied in different high dimensional scenarios. In the final part, we address two different approaches to get insight into multimedia data, in special soccer sport videos. While the first make use of multidimensional projections, the second uses efficient visual metaphor to help non-specialist users in browsing and getting insights in soccer matches.

SAMENVATTING

De voortdurende opmars van nieuwe technologieën hebben een rijk type van informatiebronnen beschikbaar gesteld voor analyse en investigatie. In deze context, multidimensionale dataanalyse is zeer belangrijk wanneer men moet handelen met grote en complexe dataverzamelingen. Tussen de mogelijkheden voor de analyse van dergelijke daa visualisatietechnieken kunnen gebruikers helpen om relevante patronen en trends te vinden en analyseren. Voorbeelden van multidimensionale dataanalyse omvatten beeldclassificatie, semantische 'word clouds', en clusteranalyses van documentverzamelingen en exploratie van multimediaverzamelingen.

Dit proefschrift presenteert verschillende visualisatiemethodes voor de interactieve exploratie van multidimensionale dataverzamelingen voor beide gespecialiseerde en gewone gebruikers, gebaseerd op statische en dynamische representaties gebouwd met multidimensionale projecties. We presenteren eerst een multidimensionale projectietechniek die afstanden goed bewaart voor alle types multidimensionale gegevens, met toepassingen in multimedia en text-document verzamelingen. Verder analyseren wij de taak van projectieinterpretatie in 2D gebaseerd op lokale fouten. Verder presenteren wij een groep interactieve visualisaties die de gebruiker in staat stelt om 3D projecties te interpreteren, voor verschillende hoogdimensionale scenario's. Als laatst presenteren wij twee aanpakken om inzicht te krijgen in multimediagegevens, zoals sportvideos, gebaseerd op multidimensionale projecties en ook bijpassende visuele metaforen om non-specialistische gebruikers in staat te stellen om voetbalwedstrijden door te bladeren om inzicht te vergaderen.

RESUMO DA TESE

O advento contínuo de novas tecnologias tem criado um tipo rico e crescente de fontes de informação disponíveis para análise e investigação. Neste contexto, a análise de dados multidimensional é consideravelmente importante quando se lida com tais conjuntos de dados grandes e complexos. Dentre as possibilidades ao analisar esse tipo de dados, a aplicação de técnicas de visualização pode auxiliar o usuário a encontrar e entender os padrões, tendências e estabelecer novas metas. Alguns exemplos de aplicações de visualização de análise de dados multidimensional vão de classificação de imagens, nuvens da palavra semântica, análise de grupos de coleção de documentos à exploração de conteúdo multimídia.

Esta tese apresenta vários métodos de visualização para explorar de forma interativa conjuntos de dados multidimensionais que visam de usuários especializados aos casuais, fazendo uso de ambas as representações estáticas e dinâmicas criadas por projeções multidimensionais. Primeiramente, apresentamos uma técnica de projeção multidimensional que preserva fielmente distância e pode lidar com qualquer tipo de dados de alta-dimensionalidade, demonstrando cenários de aplicações em ambos os casos de multimídia e coleções de documentos de texto. Em seguida, abordamos a tarefa de interpretar as projeções em 2D, calculando erros de vizinhança. Posteriormente, apresentamos um conjunto de visualizações interativas que visam ajudar os usuários com essas tarefas, revelando a qualidade de uma projeção em 3D, aplicadas em diferentes cenários de alta dimensionalidade. Na parte final, discutimos duas abordagens diferentes para obter percepções sobre dados de multimídia, em particular vídeos de futebol. Enquanto a primeira abordagem usa projeções multidimensionais, a segunda usa uma eficiente metáfora visual para auxiliar os usuários não especialistas em navegar e obter conhecimento em partidas de futebol.

CONTENTS

1 INTRODUCTION

In the last decade, the informational world has witnessed a major shift in structure, operation, and focus. This shift can be explained by the improvement and convergence of a large and hybrid number of technical developments. The advent of faster, cheaper, and easier to deploy and use sensor technologies, such as video cameras, GPS devices, medical body signal meters, proximity sensors, and laser scanners have made a rich and growing type of information sources available to analysis and investigation. The separate advent of cheap and fast storage space for data has made it possible to gather immense amounts of data of various types for the same goals of analysis and investigation. The development of web-based communication technologies such as web and cloud computing and software-as-a-service has dramatically increased both the range and bandwidth of data-processing functionality available to stakeholders ranging from large corporations to casual users.

The above developments, which are often known under the generic name of 'big data', has opened many types of applications and investigations which were, until recent times, either thought to be hardly possible, or else confined to the realm of a narrow set of technology-intensive organizations. Nowadays, big data applications such as finding best-purchased products, most-relevant news articles and blog discussions, trends and outliers in shopping or travel of large groups of individuals, key topics in healthcare and lifestyle, and the evolution of stock portfolios under various market, social, and political factors, are available to both specialized corporate professionals and end users.

However, these unprecedented developments that put data (and, implicitly, its analysis and interpretation) at the core of modern-day developments and decision-making, have also created a number of important challenges. At a high level, such challenges can be captured by the three main characteristics which are typically associated with big data, often called the big data's '3V': *volume*, *velocity*, and *variety* [156]. Volume refers to the sheer amount of data items that can be acquired and stored with relatively low cost and effort, but whose processing and interpretation requires comparatively much larger efforts. Velocity refers to the speed of change of data that is to be investigated, and is a by-product of the decreasing acquisition and storage costs, and the increasing simplicity and throughput of data sensors. As such, velocity can be seen as a multiplier for the volume factor, which in turn increases the challenges and needs for efficient, effective, and scalable analysis and interpretations techniques and tools. Finally, variety refers to the increasing range of types of data that we can acquire, store, and wish to analyze. These range from classical continuous signal measurements, such as temperature, pressure, speed, velocity, altitude, and heartbeat (delivered by typical analogue-to-digital converters and sensors) to text documents (mined from blogs, Twitter accounts, RSS feeds, and web-crawling tools); structured relational attributed graphs (mined *e.g.* from open-source software repositories such as CVS, Subversion, and Git); and multimedia data involving collections of text, metadata, video, and audio streams (mined from surveillance cameras, live TV broadcasts, or web sites such as YouTube, Facebook, or Instagram).

Critical to an efficient and effective use of big data is the availability of equally efficient and effective technologies to extract so-called *actionable insight* [120, 143]. Actionable insight can be generically defined as higher-level knowledge, extracted from 'raw' big data collections, which enables organizations to take actions that optimize activities such as sales, marketing, product optimization, and reducing cost and turnover times. Clearly, the three characterizing aspects of big data (volume, velocity, and variety) pose various challenges to extracting such actionable insight in efficient and effective manners.

Within the above-mentioned challenges, data *variety* poses several particularly hard challenges. To explain this, let us consider a large and quickly varying body of data of the same nature, such as a large stream or time-series of velocity measurements coming from a speed sensor or a real-time stream of intraday stock prices. While the volume and velocity of such data sources can clearly be very large, the analysis of the underlying phenomenon captured by the sampled data is relatively simple. Well-proven and well-understood techniques such as data resampling, aggregation, summarization, statistical models, and Fourier analysis can be all used to reduce the amount of data to manageable sizes for further automated or manual analysis, by eliminating redundancies, noise, and keeping interesting patterns. In contrast, let us consider a collection of multimedia data involving images, videos, subtitles, and other metadata such as acquisition time, type of captured content, user comments, and quality rankings, such as *e.g.* delivered by a mainstream source like YouTube or Facebook. While the amount of bytes in such a multimedia collection may be identical to the amount of bytes in one of the earlier-described examples of time series, its analysis and interpretation is considerably more challenging. One of the reasons hereof is that each data item in the aforementioned multimedia collection is characterized by a large *set* of measurements, or attributes, whereas one similar data item in the aforementioned time-series is characterized by a *single* measurement or attribute. The second reason is that the various attributes of a single data item in the multimedia dataset example all have different natures, or *types*, *e.g.* images, sound, text, and metadata annotations.

1.1 MULTIDIMENSIONAL DATA

The above-mentioned aspects of big data variety can be, technically, captured by the concept of a *multidimensional* dataset. Two elements are key to the description of such a dataset, as follows. First, an *observation* (also called sample, instance, or data point) comprises a set of measurements that pertain to the same entity. Examples are the video stream, audio stream, subtitles, textual descriptions, and metadata that jointly describe an uploaded video on YouTube or a TV broadcast of a movie or sports match; or the files, folders, commit logs, change requests, quality metrics, and bug reports that jointly characterize a software project stored in a software repository such as Subversion or Git. Secondly, as implied by the above definition, each observation is characterized by a number of measurements (also called *attributes*, variables, or dimensions). In our examples these are the characteristics of the video, audio, text, and metadata that describe an uploaded video; or the programming language, source code, and number of type of reported bugs or filed change requests that describe a software project.

In technical terms, *multidimensional* datasets can be modeled as a set of m observations, each one being regarded as point in an n-dimensional space. While elegant, generic, and compact, this data model creates several challenges from the perspective of extracting actionable insights. Arguably the largest such challenge relates to the inability of humans to depict spaces having more than 3 or, at best, 4 dimensions. Indeed, humans live and act in a three-dimensional space; as such, understanding how data behaves in spaces having considerably more dimensions is hard, if not even impossible.

1.2 VISUALIZING MULTIDIMENSIONAL DATA

One important question to consider is why we would need to depict, or visualize, such high dimensional spaces. It can be argued that, after all, one could analyze high-dimensional datasets automatically, by running appropriate data mining or pattern detection tools and techniques, without the need to explicitly visualize the high-dimensional data. While this is true, the above reasoning omits a so-called *boostrapping* problem: To be able to design an effective pattern detection or data mining algorithm, one has to *understand* which patterns are relevant for the higher-level application-dependent goals at hand. Subsequently, in order to understand the relationships between such patterns and goals, one has to be able to visualize the patterns and their underlying ground-truth evidence to detect their characteristics which can, next, be encoded into automatic data mining or pattern searching algorithms. Separately, to be able to validate, fine tune, and improve the aforementioned data mining algorithms, one has to be able to correlate their behavior with the underlying data at hand and available ground-truth evidence. As such, at the core – or better said, at the inception – of the design of data mining algorithms for multidimensional datasets is the ability of humans to directly investigate such datasets.

The direct investigation of large and complex data spaces forms the subject of specific fields of science. At a high level *data visualization* is concerned with the creation of 2D and 3D depictions of all kinds and types of data collections, or *datasets*, with a focus on datasets which are naturally represented by spatial shapes, such as 2D images, 3D volumetric scans, and the results of 2D and 3D measurements and simulations of physical processes such as heat dissipation, mechanical deformation, or fluid flow [198, 217]. At a more specific level, *information visualization* is concerned with the creation of 2D and 3D depictions of datasets which do not have a natural representation in physical space and/or whose dimensionality is far higher than three, such as graphs, networks, relational data tables, or our multidimensional datasets consisting of multimedia or software measurements introduced earlier [233, 147]. At an even more specific level, *visual analytics* refines both data and information visualization by adding customized workflows, typically based on interactive techniques, to analyze the depicted data with the aims of forming, (dis)proving, and refining hypotheses – a process globally known under the name of 'sensemaking' [219, 111]. At a global level, all above-mentioned visualization techniques have the same aims – making the invisible (data) visible, and supporting activities such as confirming the known and discovering the unknown (in the data).

Central to the design (and success) of a data visualization application is the ability of mapping the *essential* aspects of the dataset(s) under study to recognizable 2D and/or 3D shapes, colors, and patterns. If such patterns are carefully chosen, users can (a) easily recognize them, even in the presence of considerable noise and variability, and (b) next map them back to phenomena present in the original data, by a process known as 'inverse mapping' [217]. Much work has been dedicated to the study of which so-called visual encodings (of data) best serve specific data analysis tasks for specific kinds of datasets [222]. This work has given rise to a wealth of specific methods and visualization subfields such as scalar, vector, and tensor visualization [198]; graph and network drawing [56]; and information visualization [147].

In the above context, multidimensional data poses a particularly hard challenge to visualization methods. Indeed, as mentioned earlier, finding and constructing appropriate visual encodings for observations having tens or even hundreds of dimensions is hard, since our visual space accommodates just a few dimensions. The solution of mapping each separate dimension in the original dataset to a separate visual channel, such as *e.g.* shape, position, color, size, texture, and shading is at best limited to four or five dimensions, since the aforementioned visual channels interfere with each other and are also not equally suited to map all kinds of attribute types [222].

1.3 MULTIDIMENSIONAL PROJECTIONS

A different solution to the visualization of multidimensional data is proposed by the so-called *multidimensional projection* methods. These methods are also known under various other names, such as dimensionality reduction and multidimensional scaling. In a nutshell, such methods take a dataset having hundreds or even thousands of dimensions per observation, and construct a new dataset having the same number of observations but only two or three dimensions. Key to the working of projection methods is their aim to preserve the so-called *data structure* in the resulting low-dimensional space. This takes the form or *e.g.* preserving distances between pairs of observations or preserving the neighborhoods of observations. If such goals can be achieved efficiently and effectively, the users can employ the resulting low-dimensional projection as a 'proxy' to study the invisible high-dimensional spaces. For instance, tasks such as finding outliers, groups of highly-related observations, correlated dimensions, trends in the dimensions' values can all be performed on the low-dimensional space, which is typically displayed as a color-coded scatterplot of observations.

In the last decade, a wealth of multidimensional projection methods has been developed. Current methods, and their corresponding implementations, are very successful in computing projections of datasets having a large number of observations, each having tens dimensions or more, at near-interactive rates, and with a good preservation of relevant quality metrics such as distances or neighborhoods of observations. Additionally, most such methods can be readily used by developers and end users in a largely black-box fashion, *i.e.*, without having to know or understand the intricacies of their implementations. However, projection-based visualizations are significantly hampered by their abstract nature: In the typical case, they deliver a 2D scatterplot where each observation is encoded as a point. While this allows one to tell whether the dataset under study contains outliers, or groups of related observations, it does not tell users *why* such structures appear in the data. For instance, we can easily locate an outlier observation in a 2D scatterplot created by a multidimensional projection, but we do not know which attribute(s) of the respective observation make that point so different from the rest. Similarly, we may be able to easily see that a set of observations consists of three groups of well-separated points, but we do not know which attribute(s) make the points in a group belong together. Altogether, these aspects significantly decrease the usability and acceptability of projection techniques for large classes of non-specialist users such as business and market analysts, let alone casual end users.

1.4 MULTIMEDIA DATA

As outlined above, multimedia data are of considerable interest in our context of visual analysis of multidimensional datasets. Summarizing the reasons of our interest in multimedia data, we outline the following key factors:

- *Prevalence:* Multimedia data are of increasing availability and importance in all contexts where big data plays a role. It is available at low or even zero costs, being provided at high volume, velocity, and variety on social media channels such as YouTube, Facebook, the world wide web, or more specialized movie or TV collections;

- *Complexity:* Multimedia data are by excellence highly variate, in the sense of containing a (large) range of attributes of different types. These include video and image descriptors such as *e.g.* SIFT and SURF features [136], image moments [181], color and brightness histograms [202], and features extracted from face recognition [247]; audio descriptors such as pitch, volume, and speech-to-text descriptors [11]; and metadata descriptors such as keywords extracted from provided subtitles, user comments, or categorical ratings and classifications of videos. Altogether, the above provide a rich high-dimensional collection of attributes of various types that characterizes multimedia data – and therefore an explicit challenge to multidimensional data visualization;

- *Reach:* Multimedia data has a clearly wide reach to extremely various types of users being interested in highly different goals and having highly different skills and expectations. These range from professional surveillance analysts and coaches of sports teams, who have the training and time required to interpret highly detailed and sophisticated analyses, to casual end users such as teenagers browsing video collections or sports fans watching series of soccer matches. While the input data is identical in all these cases, the techniques required to satisfy the needs of these user groups clearly have to be of different types.

Of course, multimedia data are not the sole source of multidimensional dataset that poses related challenges in terms of designing efficient and effective visual exploration methods. Similar data sources exist in the form of *e.g.* software repositories, statistics delivered by web crawlers, patient medical records, or stock transactions. However, we argue that all three above-mentioned aspects – prevalence, complexity, and reach – are considerably higher for multimedia data. Indeed, multimedia data is available in larger amounts (and varieties), than open-source software projects or the more heavily controlled patient records or paid stock streaming data; it contains attributes of a wider category (*e.g.* image, audio, and text descriptors) than the typically more similar attributes of software systems (relations, quality metrics, and text documents) or stock data (measurements typically involving only scalar quantities such as stock open, close, volume, and bid/ask prices); and it involves both professionals and casual users, while software, medical, and stock data are mainly the object of analysis of dedicated professionals. As such, multimedia data forms an interesting target for the development of novel methods for the visual exploration of multidimensional data.

1.5 RESEARCH QUESTIONS

Having introduced the generic challenges of visual exploration of multidimensional data, and the more specific challenges of exploring multimedia data, we can now formulate our two key research questions:

Question 1: *How can we design ways to interactively explore multidimensional projections that convey to users insights on the semantics of the patterns perceived in the projection space, in terms of aspects of the high-dimensional data?*

Question 2: *How can we design ways to interactively explore multidimensional data extracted from multimedia datasets so as to support a wide range of tasks for different types of users ranging from professionals to casual users?*

The above two questions are related at several levels. First, effective and efficient solutions to Question 1 may serve the basis of designing effective and efficient solutions (tools) that satisfy the goals posed by the users listed under Question 2. Secondly, specific goals and requirements of the users under Question 2 can form test scenarios to validate, or show the limitations of, the methods designed to solve Question 1. Thirdly, multimedia data forms by itself a rich, complex, and easily available corpus of information that serves to test methods designed to solve both Questions 1 and 2.

In line with the above two coupled research questions, the structure of this thesis is as follows.

In Chapter 2, we discuss related work in the areas of (visual) analysis of multidimensional data and the more specific analysis goals of multimedia data.

Chapter 3 explores the use of projection techniques for the visual depiction of large multidimensional datasets. To this end, we present a novel multidimensional projection technique which competes favorably, or even exceeds, desirable features of state-of-the-art projection techniques such as generality, computational scalability, precision in preserving inter-point distances, algorithmic robustness, and – last but not least – the ability to control the shape of the resulting projection by interactive placement of a small number of selected data points. We demonstrate the application of our proposed technique by two use cases involving the joint exploration of audio-and-video multimedia collections and the exploration of collections of text documents.

Chapter 4 addresses the task of interpreting projections created by typical multidimensional techniques, such as the technique introduced in Chapter 3, from the perspective of understanding how the projection itself and its errors, in terms of depicting distances in the high-dimensional data structure, are affected by parameters of the projection techniques being used. We introduce several metrics to quantify the quality and variability of a projection, and show how such metrics can be visually depicted in intuitive and easily usable ways. By this, we make the first step into explaining multidimensional projections to their typical end users. We demonstrate our proposed techniques for a wide range of multidimensional datasets and existing projection techniques.

Chapter 5 addresses the task of explaining 3D multidimensional projections in terms of the attributes, or dimensions, of the projected high-dimensional datasets. Specifically, we present a number of interactive exploration and explanation mechanisms that inform users on the meaning of the visible data structures in a 3D projection in terms of the underlying high-dimensional variables, and let users browse the space of possible viewpoints of a 3D projection to find viewpoints from which specific variable-groups can be best analyzed. This makes the second step in our quest towards explaining multidimensional projections to their typical end users. We demonstrate our proposed techniques for a range of multidimensional datasets and projection techniques, and also show the added-value of using 3D projections, annotated by our explanatory mechanisms, as compared to the better-known 2D projections.

Chapter 6 goes back to our second research question – the exploration of multimedia datasets. We show how, for a particular category of such datasets – sports videos with additional metadata – we can use multidimensional projections to extract a number of high-level insights in the structure and dynamics of the underlying sports games. Such techniques typically address the segment of more professional users such as analysts and sports coaches. Additionally, we present a diametrically opposed approach to the visual exploration of sports videos collections aimed at casual end users (sports fans) that does not use multidimensional projections. By correlating the insights gathered by the two presented visualizations, we outline strengths and limitations of the use of multidimensional projections in visually exploring multimedia datasets.

2 RELATED WORK

ABSTRACT: *Multidimensional datasets pose numerous challenges in terms of efficiently and effectively analyzing them to extract useful and usable insights for problem solving. Several of these challenges stem from the difficulty of capturing and explaining patterns caused by the values of multiple attributes sampled over many observations, and from the fact that it is hard for humans to form an intuitive depiction of high-dimensional data spaces. In this chapter we present a taxonomy of multidimensional data and overview the different analysis and visualization methods proposed for exploring such data, with a focus on multidimensional projection methods. Separately, we overview existing methods for the visual exploration of large multimedia data collections, with a focus on the multidimensional nature of the involved data. We conclude that multidimensional projections can be efficient and effective techniques to use in the construction of visual exploration tools for multimedia datasets.*

2.1 INTRODUCTION

Given the technological advances over the years, generation and consumption of data by content producer companies, and also by domestic users, has witnessed a continuous growth. In this context, analysis concerns the process starting by obtaining batches of (raw) data and converting these into information useful for decision-making by various types of users. During this process, data are collected and analyzed with the aims of answering questions; creating, testing, and refining hypotheses or to (dis)prove theories; or presenting findings and insights to the interested public [107]. In this chapter, we discuss techniques related to the above activities which treat multidimensional data, our focus of research.

2.2 MULTIDIMENSIONAL DATA IN CONTEXT

Before starting the analysis process, it is essential to have a good understanding of the kind of data that is being analyzed. Therefore, the first step in the analysis of data is to get clarity about the data's intrinsic nature, meaning, structure, and type. Characterizing the nature and meaning of data by, for instance, capturing the variability of these aspects into a classification or taxonomy, is a very challenging endeavor [147]. Indeed, the same dataset can be regarded from multiple perspectives, depending on the type of questions or analysis that we want to address. As such, while having a good understanding of data nature and meaning is definitely important for solving a concrete data-related problem, there are few universal guidelines to be applied in this process. In contrast, characterizing the data *structure* and *type* by means of taxonomies is a useful and effective instrument to guide researchers towards specific classes of analysis and visualization techniques suited for their concrete datasets [198, 204].

2.2.1 *Taxonomies of data*

Several taxonomies have been proposed for classifying data in terms of structure and type. With respect to the visual exploration (or visualization) options that such taxonomies associate to different datasets, we note the classification of data in terms of *spatial vs non-spatial* [147]: *Spatial* data is seen as the value of a function f whose domain D is a (typically compact) subset of $\mathbb{R}^2$ or $\mathbb{R}^3$. Data values, or samples $f(\mathbf{x}_i)$, are recorded at sample points $\mathbf{x}_i \in D$. Depending on the distribution of the sample points $\mathbf{x}_i$ over D, we distinguish between different types of dataset sampling strategies, or *grids*, such as uniform or regular, rectilinear, structured, and unstructured [217]. Different types of grids trade off implementation simplicity and low storage costs against flexibility to place sample points at desired locations over D to achieve an optimal capture of the shape of f with a minimal number of samples. *Non-spatial* data is seen as the value of a function f whose domain D is not a (compact) subset of a continuous space such as $\mathbb{R}^2$ or $\mathbb{R}^3$. Implementation-wise, non-spatial data also consists of a number of data values $f(\mathbf{x}_i)$ recorded at a number of points $\mathbf{x}_i \in D$. However, in contrast to spatial data, the points $\mathbf{x}_i$ can not be regarded as a *sampling* of a continuous space – there is simply no data between the points $\mathbf{x}_i$. As such, the points $\mathbf{x}_i$ where data is recorded are typically called data points or observations, rather than sample points.–

A further refinement of spatial data taxonomy regards the type of values of the function f, or the range R of f. Such values $f(\mathbf{x}_i)$ are also called data *attributes*. Attributes are typically classified according to the operations that R permits. In decreasing order of sophistication, the following attribute types are commonly identified [147, 217, 86]:[1]

- *quantitative:* Quantitative attributes, also called continuous or ratio attributes, are defined over ranges R that allow operations such as addition, subtraction, and multiplication by a real number. Most commonly, such attributes are real values, *i.e.* $R \subset \mathbb{R}$. Spatial data having quantitative attributes is frequently met in the context of so-called *scientific visualization* (scivis) datasets. Examples hereof are height fields, grayscale or color images, CT or MRI scans, and 2D and 3D vector fields created by computational flow dynamic (CFD) simulations [86]. A key characteristic of spatial datasets having quantitative attributes is that these datasets naturally allow *interpolation* of data values. In detail, for any point $\mathbf{x}$ located in the dataset domain

1 Attribute taxonomies bear an interesting, though not further exploited similarity (to our knowledge), with the taxonomy of multivariate ordering proposed long ago by Barnett [12]. In this taxonomy, multivariate attributes are classified into those admitting marginal, reduced (aggregate), partial, and sequential (categorical) ordering.

D, we can estimate the interpolated value of the function $f(\mathbf{x})$ as a function of the data values, or samples, $f(\mathbf{x}_i)$ at sample points $\mathbf{x}_i \in D$ located at points $\mathbf{x}_i$ close to $\mathbf{x}$. Interpolation is a crucial capability for supporting operations such as reconstruction (re-creating a piecewise continuous, version of f over D from the samples $f(\mathbf{x}_i)$), resampling, and smoothing. In turn, such operations address tasks such as data simplification and aggregation, which support the visual exploration of very large datasets.

- *integral:* Integral attributes, sometimes also called discrete attributes, are defined over ranges R that allow operations such as addition and subtraction, but not multiplication by a real number. Most commonly, such attributes are domains $R \subset \mathbb{Z}$. Non-spatial data having integral attributes is frequently met in the context of so-called *information visualization* (infovis) datasets. Examples hereof are tables whose cells represent counts, like number of persons in a census [147]. In contrast to quantitative datasets, integral datasets do not (formally speaking) admit interpolation. Indeed, even if this is technically possible, interpolation of integral values would typically create real values, which thus are outside of the domain $R \subset \mathbb{Z}$.

- *ordinal:* Ordinal attributes are defined over ranges R that allow operations such as ordering, *i.e.* define the relations $<$, $>$, and $=$. Examples of such attributes are ordered sequences of ranks of observations, such as Likert 5-point scales used to assess the quality of a product, *e.g.* $R = \{$*very poor, poor, neutral, good, very good*$\}$, or scales used to quantify the acceptance likelihood of a scientific publication submitted for review, *e.g.* $R = \{$*definitely reject, possibly reject, bordeline, possibly accept, definitely accept*$\}$. Ordinal attributes typically do not allow interpolation.

- *categorical:* Categorical, or nominal, attributes are defined over any set R. The only operation allowed by such attributes is, thus, checking for identity or equality of two elements. Examples of such attributes are types of elements, such as gender (male or female) or vehicle type (car, plane, train, or ship). Categorical attributes can be further organized in hierarchies or taxonomies, based on the perceived similarity of data values. When this is possible, categorical attributes also allow computing the distance, or similarity, between data values. As distance is an integral or quantitative value type, this allows mapping categorical attributes to data types that allow more powerful operations, thus support a wider range of analyses and explorations.

- *text:* Text attributes are defined over the set of all possible text strings generated by a given letter alphabet, or over a more restricted set of phrases or words captured by a given dictionary. Formally speaking, text attributes can be seen as either ordinal (since strings admit ordering *e.g.* in terms of lexicographic order) or categorical (since we can easily tell when two strings are identical or not). Moreover, distances can be computed over text attributes, *e.g.* in terms of the Levenshtein metric [128]. However, in many practical applications, text attributes attempt to capture more involved information than what string comparison, lexicographic order, and Levenshtein distances can model. For instance, typical text analysis and mining applications need to avail of more complex metrics that capture the semantic similarity of text fragments. When such metrics can be computed from text attributes, we can reduce such attributes to (sets of) quantitative, ordinal, and categorical derived attributes.

- *relational:* Relational attributes are defined over sets of data points in D. Their range R is thus the set of all possible subsets of elements in D, or the power set of D. In many fields, such attributes are known as graphs or networks. Here, the nodes represent data points in D, and edges capture the relations between these data points which are part of R. Graphs can be further specialized into directed and undirected, cyclic or acyclic, and trees or hierachies. Graphs are ubiquitous in many information visualization subfields, such as *software visualization* (softvis), where they capture the structure and dependencies of software systems [57]; and *geographical visualization* (geovis) [62], where they capture the structure of road or similar transportation networks. The visual exploration of graphs forms the focus of the specialized subfields of graph visualization and graph drawing [56]. Relational attributes form a separate case as compared to the earlier discussed attribute types (quantitative, integral, ordinal, categorical, and text): Indeed, while these earlier attribute types describe a property *solely* associated to a data point or observation $\mathbf{x}_i$, relational attributes describe properties associated to *sets* of (minimally two) such data points $\{\mathbf{x}_i\}$.

The above hierarchy of attribute types offers a nesting in terms of capabilities: Quantitative attributes are also integral; integral attributes are also ordinal; and ordinal attributes are also categorical. Text attributes can be reduced to combinations of quantitative, ordinal, and categorical attributes. Relational attributes form a separate class, for which special analysis and visualization methods have been designed, and which are known under the generic name of graph drawing (or graph visualization) techniques [230, 56, 88]. Multidimensional datasets including relational attributes forms a sub-field of interest of graph visualization, which proposes specific methods that aim to emphasize the relational nature of the data and also visually encode several attributes per node and/or edge [58, 234, 180]. Overall, the nesting of atribute types indicated above allows using visual exploration and analysis methods defined for the less 'powerful' attribute types to be applied to more powerful attribute types, but not conversely.

Datasets having quantitative attributes defined at data points sampled over spatial domains are sometimes also called *continuous* datasets, as they allow interpolation, as explained earlier. In contrast, datasers having (1) quantitative attributes defined over non-spatial domains, and also (2) datasets having any attributes besides quantitative ones, are sometimes called *discrete* datasets, as they do not allow interpolation, either because of the lack of a distance metric between sample points (case (1) above), or because of the lack of necessary operations for interpolation such as multiplication with a real-valued number (case (2) above). To strengthen the difference between continuous and discrete datasets, the latter are sometimes also called inherently discrete datasets [217]. This underlines the difference between a discrete dataset containing quantitative attributes, obtained by the sampling of a continuous signal over a spatial domain, which naturally admits interpolation to a piecewise-continuous result; and a discrete dataset of types (1) or (2), which does not admit interpolation, for the reasons outlined above.

Other taxonomies of, or related to, data used in visualization have been proposed. For instance, Shneiderman proposes a taxonomy of visual exploration tasks by the data type being involved in the respective task [204]. Seven data types are recognized: one-, two-, and three-dimensional datasets (the dimension here being roughly equivalent to the dimensionality of our set D introduced at the beginning of Sec. 2.2.1); temporal datasets; trees; networks; and multidimensional data. However, the taxonomy is not further refined in depth up to a level where one can decide on visualization methods best suited for a given data type. Also, multidimensional data is discussed only briefly. Chan proposes a taxonomy of visualization techniques for multivariate data, along the lines proposed by Keim and Kriegel [114, 112] in geometric, icon-based, pixel-oriented, hierarchical, graph-based and hybrid techniques. However, this work does not outline an explicit taxonomy for multidimensional data itself.

2.2.2 *Multidimensional data*

Following the dataset model presented above (a function $f : D \to R$), we can further distinguish between datasets which record a single-valued attribute per point and datasets which record multi-valued attributes per point. Examples of the first category, for spatial datasets, are scalar-valued datasets or scalar fields ($R \subset \mathbb{R}$); 2D and 3D vector-valued datasets or vector fields ($R \subset \mathbb{R}^2$ and $R \subset \mathbb{R}^3$, respectively); and color images ($R \subset \mathbb{R}^3_+$). If we generalize this idea, we obtain the case of multidimensional datasets where $R \subset \mathbb{R}^n$, where each data point has n real-valued attributes. Examples hereof include numerical simulations where, at each data point, one records several physical quantities, such as velocity, pressure, temperature, and matter density.

Multidimensional datasets are also ubiquitous in the case of non-spatial datasets. Probably the best known example of such datasets are *data tables*. In a data table, we can see each row as a data point or observation $\mathbf{x}_i$. Each column of the table, thus, records the values of a different attribute over all observations. Hence, a table having m rows and n columns records m observations each having n attributes, or an n-dimensional attribute [204]. In our functional notation introduced in Sec. 2.2.1, such a data table can be thought as a function $f : \{1, \ldots, m\} \rightarrow R^n$, where R is the domain of definition of the values of a data table cell.

Variable types: Within the realm of multidimensional data, we can distinguish two subcases, based on the existing dependency relations between attributes. Consider, for instance, a m-column data table where the last $k < m$ columns represent the output of a simulation and the first $m - k$ columns represent the corresponding simulation inputs. We say, in other words, that the last k columns *depend* on the first $m - k$ columns. This allows classifying attributes, also called variables, into dependent and independent ones. Related to this, traditional statistics refers to attributes as variates, with their complexity associated with univariate, bivariate and multivariate data, as a function on the number k of dependent variables [82]. In the visualization field, however, the terms 'multidimensional' and 'multivariate' are often used interchangeably to denote a dataset where, for each data point or observation, we have more than one attribute value (thus, $m > 1$). For instance, some authors relate the above two terms to the dependent *vs* independent nature of attributes, by using the term 'multivariate' when we have several dependent variables ($k > 1$) and 'multidimensional' when we have several independent variables ($m - k > 1$) [60]. In contrast, other authors use the terms 'multivariate' [217], 'multidimensional' [110, 204], or 'multivariables' [60] to refer to both independent and dependent variables. As a consequence of the above terminology, values of observations are also known under different names, such as data-point values, observation values, dimensions, attributes, variables, and features [147].

Datasets, fields, dimensions, variables: Complicating the above discussion on terminology, datasets consisting of several observations, each having multiple scalar attributes, are also known under the names of *vector-valued data* and *vector fields*. Although formal and universally-accepted distinctions between these terms is not yet present, the respective terms represent different situations. Vector-valued data is a term typically used to mean the same as our earlier introduced terms of multidimensional (or multivariate) data – that is, a dataset where each sample or observation is described by several scalar-valued attributes. Vector fields, in contrast, are a subset of vector-valued data, where, at each point or observation $\mathbf{x} \in R$ in the domain of definition, we can define a so-called *tabgent vector*, *i.e.* a vector that leads us from $\mathbf{x}$ to another point located *inside* R. Examples of vector fields are color images ($m = 3$ scalar attributes, $R \subset \mathbb{R}^2$) and flow fields in three-dimensional computational flow dynamics ($m = 3$, $R \subset \mathbb{R}^3$). In contrast, a general data table having three columns represents, technically speaking, the same dataset ($m = 3$ attributes), but is not a vector field, since there is no domain R having the aforementioned tangent vector property. A separate distinction between vector-valued data and multivariate data refers to the *interpretation* of the data attributes. In vector-valued datasets, the m attributes recorded per observation *may* be related, but can be studied equally well independently on each other, like in the case of a data table recording the age, salary, and profession of a set of individuals. In contrast, in a multivariate dataset, the same m attributes are *intrinsically* related, so they should be (normally) studied together, like in the case of an image recording the red, green, and blue components of each pixel.

As the focus of the work in this thesis concerns the analysis of data having multiple attribute values per observation ($m > 1$), but without making a specific distinction between dependent and independent variables, and without making any assumptions on the nature of the domain of definition R of these variables, nor on the existing and/or important relationships between individual attributes, the terminology we have adopted is that of *multidimensional data*. This terminology is the more common one being found in information visualization literature, and it also reflects the names of several important visualization methods in our context (*e.g.*, multidimensional projections). Separately, we will next use the terms variables, attributes, and dimensions to refer to the values of observations, in line with the best suited term for the specific discussion context. Finally, when mentioning multidimensional data, our focus will implicitly be on datasets having a *high* number of dimensions per observation (tens up to thousands), rather than multidimensional data having a low number of dimensions, such as 2D or 3D vector fields.

2.2.3 *Multidimensional data analysis challenges*

Multidimensional data offers some of the largest challenges to data mining, data analysis, and data exploration, and is one of the topics of the so-called 'grand challenges' in information visualization and visual analytics [219, 111]. The difficulty of understanding (phenomena described by) multidimensional data is caused by several aspects:

- *Complexity:* Elements of interest for identification and analysis, such as patterns, trends, outliers, and clusters, are much harder to describe, define, quantify, find, and present in the case of multidimensional data than in the case of low-dimensional data such as scalar or 2D or 3D vector fields. The key reason for this is that an increased number of *independent* dimensions allows for the appearance of a considerably higher variability than a low number of dimensions. To provide a simple analogy, consider a function of a single real-valued variable $f : \mathbb{R} \rightarrow \mathbb{R}$. Analyzing such a function is relatively easy, by using *e.g.* its first and second-order derivatives to *e.g.* reason about its rate of change and local extrema. In contrast, consider a function of ten real-valued variables $f : \mathbb{R}^{10} \rightarrow \mathbb{R}$. Analyzing such a function is considerably more complex, as there exist a much larger family of first and second-order partial derivatives.

- *Abstract nature:* Low-dimensional data such as 2D or 3D fields can be relatively easy understood by directly plotting the recorded values by using a range of classical visualization methods. For discrete datasets, these methods include scatterplots, height plots, bar charts, and histograms. For data admitting interpolation, other specialized methods exist, such as height plots, contour plots, streamlines, and hedgehog plots. Such methods are well known and well proven in the domain of scientific visualization [86, 198, 217]. The key advantage of being able to directly plot low-dimensional data is that we can identify and reason about complex patterns by simply seeing them, and without needing potentially complex ways to automatically find and quantify them. For instance, it is relatively easy for moderately-trained end users to spot the presence of vortices, sources, and sinks in 2D vector fields, even though automatic detection thereof is still a complex problem [117]. In contrast, humans do not have a direct intuition of spaces of dimensionality higher than 3. As such, directly understanding high-dimensional datasets is considerably harder, as we have to somehow *project* them into the low-dimensional (2D or 3D) spaces we are able to see and reason about.
- *Discontinuous nature:* Many multidimensional datasets involve data attributes which do not (easily) admit interpolation. Ordinal and categorical attributes are prime examples hereof (Sec. 2.2.1). As such, we cannot create smooth, continuous, representations of such datasets with the same ease as we can, for example, when dealing with quantitative attributes. This creates several challenges even for low-dimensional data. For example, consider a 3D scalar volume, such as a CT or MRI scan. As the underlying data from which this dataset was constructed (via sampling) is inherently continuous, we can use various interpolation techniques to create smooth, continuous, and easy to understand representations thereof, such as volume-rendered visualizations [133]. In contrast, consider a data table having three columns, all containing quantitative attributes. We can visualize these data by means of a 3D scatterplot, which would generate a 3D point cloud. As there is no continuity (no information is plotted between the points), understanding such a point cloud is much harder than understanding the earlier 3D scalar volume field.

Let us remark separately that the challenges of understanding multidimensional data are not first and foremost related to the *amount* of data being involved. Indeed, consider a 1D scalar dataset having one million sample points which record the samples of a function $f : \mathbb{R} \rightarrow \mathbb{R}$ We can easily construct a view of such a dataset, *e.g.* in terms of a classical graph $y = f(x)$, and explore the graph by classical interaction techniques such as zooming and panning. In contrast, consider a 100-dimensional scalar dataset having 10000 sample points. While the amount of data are the same as for the first case, understanding this 100-dimensional dataset is considerably harder. This is due mainly to the fact that, for this dataset, understanding a *single* data point means reasoning about 100 values. In contrast, understanding a single data point for the earlier 1D dataset involves understanding a *single* data value. The problem compounds itself when aiming to perform more complex analyses. For instance, finding the distance between two data points involves, in the 1D scalar dataset case, comparing two data values; doing the same for the 100-dimensional dataset involves comparing 200 data values. This inherent problem caused by the number of attributes in multidimensional data is sometimes referred to as the 'curse of dimensionality' [16, 244]. Such understanding challenges involving high-dimensional datasets will be discussed in more detail in Sec. 2.4 in the context of visual data exploration.

2.3 MULTIDIMENSIONAL DATA ANALYSIS

Given our focus on multidimensional data exploration, we next discuss methods and techniques aimed to support various types of analyses of such data. These techniques will form the basis of the visual exploration, or visualization, techniques for multidimensional data discussed next in Sec. 2.4.

To unify the discussion, we first introduce several notations to refine the description, first introduced in Sec. 2.2, of a multidimensional dataset. We model such a dataset as a collection $\mathcal{D} = \{\mathbf{x}_i\}$ of m data points, or observations $\mathbf{x}_i$, which are identified by their index of ID $1 \leqslant i \leqslant m$. Each observation $\mathbf{x}_i$ is a tuple $\mathbf{x}_i = (x_i^1, \ldots, x_i^n)$ of n attributes, or variables x_i^j, $1 \leqslant j \leqslant n$. The number of variables n gives the dimensionality of $\mathcal{D}$. We denote the values of the j^{th} variable over all m points of $\mathcal{D}$ by the vector $\mathbf{x}^j = (x_1^j, \ldots, x_n^j)$. We further assume that all elements of any $\mathbf{x}^j$ belong to the same domain – or, in other words, that all n variables have well-defined types. These can be, formally speaking, any of the types discussed earlier in Sec. 2.2.1, *i.e.*, quantitative, integral, ordinal, categorical, text, or relations. However, in our discussion (and the remainder of this thesis) we will focus mainly on quantitative, integral, ordinal, and categorical attributes, as these types largely cover most typical applications involving multidimensional data. In particular, we do not assume that all attributes are only of the quantitative type, which admits the useful property of interpolation (Sec. 2.2.1). Separately, we do not assume that different variables have the same attribute types.

Data Mining: As we live in a world where vast amounts of data are generated and collected daily, either by private companies, governments or even casual users, it is evident that we need mechanisms and tools to extract useful information from this data explosion. In this context, *data mining*, also known as Knowledge Discovery from Data, or KDD for short, is a rapidly growing research-and-applications domain. The aims of KDD techniques and tools are the discovery of useful information in large and complex datasets; the identification of novel and useful patterns that – without the application of KDD methods – would otherwise remain unknown; and the creation of prediction models to discover trends in those datasets [161]. KDD techniques and tools span the entire spectrum of user involvement, ranging from fully automatic to user-supervised techniques and finally ending with techniques where the data exploration is controlled and driven in detail by the user.

Data mining processes are typically integrated in *pipelines*, or sets of cascaded operations, where the raw input data is gradually transformed to yield the extracted knowledge of interest at the end. Such preprocessing operations have been classified by Han *et al.* in the following four categories [83]:

- *Data cleaning* aims to remove noise and inconsistencies present in the input data. In our terminology, this step aims to make the dataset $\mathcal{D}$ a faithful representation of the underlying phenomenon it tries to sample;
- *Data integration* merges data from multiple acquisition sources into a coherent dataset. This involves, for instance, joining variables $\mathbf{x}^j$ acquired from different sources and/or by different sampling processes to create the final set of dimensions that characterizes $\mathcal{D}$.
- *Data transformation* transforms or consolidates data to support the application of data mining techniques. Transformation involves, for instance, filtering and resampling of the various dimensions $\mathbf{x}^j$.
- *Data reduction* reduces data size. This can take the form of reducing the number of observations $\mathbf{x}_i$ or the number of dimensions $\mathbf{x}^j$. In both cases, data reduction aims to increase the scalability of the KDD methods which are subsequently applied to the input dataset $\mathcal{D}$.

As already implied by the pipeline concept mentioned above, the above four types of data preprocessing operations are not mutually exclusive, but can be applied jointly in the analysis of specific datasets. Separately, we note the similarity of such data processing pipelines with the well-known concept of visualization pipelines used to describe data visualization processes [198, 217]: The transformation of data from its raw input form into the high-level results, or insights, delivered by the KDD pipeline, is similar to the effect of the data importing, filtering, and mapping steps of the visualization pipeline.

As the scope of this thesis is about multidimensional data, the *data reduction* preprocessing techniques mentioned above are clearly in our focus. As mentioned above, data size can be reduced by either reducing the number of dimensions $\mathbf{x}^j$ or the number of observations $\mathbf{x}_i$. The first type of technique – dimensionality reduction – will be extensively discussed in Section 2.4, given its strong integration with data visualization techniques. The second type of technique – data aggregation – is discussed next.

In our context, we denote by data aggregation the (wide) spectrum of techniques that attempt to reduce the number of observations $\mathbf{x}_i$ present in a dataset

$\mathcal{D}$, to create a reduced dataset $\mathcal{D}'$ which preserves the essence of the phenomena of interest captured by the original dataset $\mathcal{D}$ [161]. In this sense, data aggregation can be seen as a process of eliminating observations by finding similarly-valued observations $\mathbf{x}_i \in \mathcal{D}$ and next replacing subsets of such similar-value observations $\{\mathbf{x}_i\}$ by a single subsample, or representative $\mathbf{x}'_i$. This way, undesirable aspects captured by $\mathcal{D}$, such as outliers or acquisition noise, will be eliminated. Separately, the resulting aggregated dataset $\mathcal{D}'$ will now capture only the stable, statistically-relevant, structures present in the phenomenon sampled by the input dataset $\mathcal{D}$. A final advantage of data aggregation techniques is that they reduce the amount of data that needs to be treated – either by subsequent data-analysis techniques or otherwise by data visualization techniques – and thereby increase the scalability of KDD pipelines. However, data aggregation techniques come with an important (implicit) challenge: By eliminating observations, such techniques may eliminate actual features of interest from the underlying phenomenon. A well-known instance of this problem is the elimination of small-scale variations in the data, which can be seen as either noise or small-scale detail, depending on their context.

Many data mining techniques have proven to be very successful in the context of analyzing multidimensional datasets. Following Wang *et al.* [231] and Pang *et al.* [161], we identify four types of such techniques: frequent pattern mining, clustering, classification, and statistical analysis. These four types of techniques are discussed next.

2.3.1 *Statistical Approaches*

Statistical approaches are arguably the oldest and simplest types of data analysis techniques for multidimensional datasets. Many such techniques essentially aim to reduce the size and/or complexity of a given input dataset $\mathcal{D}$ to a compact representation entailing a few figures that characterize the involved dimensions $\mathbf{x}^j$. In this respect, several techniques can be applied [161]. Means or medians can be calculated for each dimension $\mathbf{x}^j$ if one wants to characterize each such dimension by a single scalar value representing a measure of the likelihood of the observations $\mathbf{x}_i$. For instance, the mean, or centroid, of the observations $\mathbf{x}_i \in \mathcal{D} \subset \mathbb{R}^n$ is given by

$$\overline{\mathbf{X}} = (\overline{x}_1, \ldots, \overline{x}_n) \in \mathbb{R}^n, \tag{2.1}$$

where $\overline{x}_j$ is the mean of dimension $\mathbf{x}^j$ of $\mathcal{D}$. Besides the mean $\overline{\mathbf{x}}_j$ of a dimension $\mathbf{x}^j$, other statistical measures can be computed to indicate how its values x_i^j vary. For instance, the *range* $r_j \in \mathbb{R}$ of a dimension $\mathbf{x}^j$ is defined as

$$r_j = \max_i(x_i^j) - \min_i(x_i^j) = \max_{i,k} \|x_i^j - x_k^j\|. \tag{2.2}$$

The range $\mathbf{r}_j$ of a dimension $\mathbf{x}^j$ identifies the interval of values that the respective dimension is spread over. For a multidimensional dataset, the tuple $= (\mathbf{r}_1, \ldots \mathbf{r}_n) \in \mathbb{R}^n$ formed by the ranges of all its n dimensions provides, thus, an axis-aligned bounding-box that includes all observations of our dataset $\mathcal{D}$. For datasets whose sampling does not include (too far-away) outliers, ranges and bounding-boxes are simple and effective ways to describe the spread of values of their observations. For datasets containing such outliers, or for situations where the sampling is highly non-uniform in terms of spatial distribution of the observations, computing the *variance* of the involved dimensions is preferred to ranges. The variance of a dimension $\mathbf{x}^j$, denoted typically as s_j^2, is defined as

$$s_j^2 = \frac{1}{m-1} \sum_{i=1}^{m} (x_i^j - \bar{x}_j)^2. \tag{2.3}$$

Since the computation of variances involved computing mean values ($\bar{x}_j$ in Eqn. 2.3), and mean computation can be distorted by outliers, the variance values s_j^2 are also sensitive to outliers, up to a certain level. To address this sensitivity to outliers, more robust statistical measures have been proposed, such as the absolute average deviation, median absolute deviation, and interquartile range [161].

However, means, ranges, and variances only describe individual dimensions $\mathbf{x}^j$ of a dataset. In nearly all contexts where multidimensional data is involved, one is interested in finding how different dimensions $\mathbf{x}^j$ and $\mathbf{x}^k$, $k \neq j$, relate to each other. This is captured by the *covariance* of two dimensions $\mathbf{x}^j$ and $\mathbf{x}^k$, defined as

$$s_{jk} = \frac{1}{m-1} \sum_{i=1}^{m} (x_i^j - \bar{x}_j)(x_i^k - \bar{x}_k). \tag{2.4}$$

The matrix $S = (s_{jk}), 1 \leqslant j \leqslant n, 1 \leqslant k \leqslant n$ encodes the covariances of all pairs of dimensions present in a dataset, including the variances $s_{jj} = s_j^2$ of the individual dimensions on its diagonal. Similar to covariance, one can compute the *correlation* of two variables $\mathbf{x}^j$ and $\mathbf{x}^k$ as

$$r_{jk} = \frac{s_{jk}}{s_j s_k}. \tag{2.5}$$

Similarly to the covariance matrix S, we can now define a correlation matrix $R = (r_{jk})$, $1 \leqslant j \leqslant n, 1 \leqslant k \leqslant n$. Diagonal entries r_{jj} indicate the correlation of a variable $\mathbf{x}^j$ with itself, which is always one. Off-diagonal entries indicate the correlations r_{jk} of different variables $j \neq k$, which take values between 1 (perfectly correlated) and -1 (perfectly inversely correlated).

Besides the above, relatively simple, statistical descriptions of multidimensional data in terms of means, medians, ranges, covariances, and correlations, more advanced methods exist. These include, among others, Support Vector Regressions (SVR) [207] and Gaussian Process Models (GPM) [183, 55]. As an example of using such methods, HyperMoVal is a tool that uses SVR to validate regression models against multidimensional data, highlighting differences between them and allowing the addition of more model parameters to refine their regression to an acceptable level of accuracy [177]. While such more complex statistical techniques can capture patterns and trends present in the data in more accurate ways than classical statistical tools, they are also more computationally involved, and also harder to explain to a wide range of end users.

2.3.2 *Frequent Pattern Mining*

Frequent pattern mining is the process of searching recurring relationships, such as associations and correlations, in a given dataset. Frequent pattern mining techniques can be further classified into techniques that search for frequent association rules and techniques that search for frequent sets of items, or itemsets. Given a dataset $\mathcal{D}$, frequent association-rule techniques search for correlations between specific values of specific dimensions $\mathbf{x}^j$ over many observations $\mathbf{x}_i \in \mathcal{D}$. For a similar dataset, frequent itemset techniques search for sets of observations $\mathbf{x}_i \in \mathbf{D}$ that are involved together in the description of patterns such as transactions in a transactional database [3].

Frequent pattern mining methods are good at discovering unknown correlations between dimensions and observations in a given dataset, when one has little or even no prior knowledge over what to search for in the data. On the other hand, these techniques are also challenged by large multidimensional datasets: As the search space for patterns grows exponentially with the number of observations and/or dimensions, heuristics are needed to limit the search to a subspace that is (a) small enough to be computationally feasible, and (b) represents well the most likely area where relevant search results are to be found. In the last decade, many such search heuristics have been developed. For example, Pasquier *et al.* proposed to mine a selective subset of frequent patterns based on the number of occurrences of a pattern is the same to all its immediate patterns, calling this method of *closed frequent patterns* [162]. The CLOSET algorithm further expedites the mining of closed frequent patterns using a frequent pattern tree as a compact representation to organize the dataset, performing a depth-first search [170]. A limitation of CLOSET is that only applicable to datasets having a low to moderately-large dimensionality – for datasets having more than $m = 100$ dimensions, efficiency starts to be compromised. To solve this high dimensionality problem, Carpenter *et al.* first transposes the matrix representing a dataset and then performs a depth-first row wise enumeration on the transposed matrix. As a result the computational cost decrease significantly [160]. A detailed survey of frequent pattern mining algorithms and their challenges is given in [77].

Frequent pattern mining methods are effective in the analysis of multidimensional datasets such as data tables that represent transactions over sets of items stored in relational databases. In such cases, observations are typically *exact* records involving integral and categorical attributes. However, many other multidimensional datasets do not fall into this class – consider, *e.g.*, observations whose dimensions are real-valued quantities measured with a finite precision, such as features extracted from color images, used next to describe the semantics of these images. For such datasets, other data analysis methods are better, as discussed next.

2.3.3 *Classification*

Generally put, given a dataset $\mathcal{D}$ of observations $\mathbf{x}_i$, *classification* is a process that aims to assign a class value, also called a *label*, to each observation $\mathbf{x}_i \in \mathcal{D}$. Class labels are taken from a typically small set of domain-specific values that describe the dataset at a high semantic level. The process of classification, therefore, aims to add this high-level semantic (class) information to a dataset $\mathcal{D}$ that contains only lower-level attributes. In the high dimensionality context, Wang *et al.* made the following suitable definition of classification: "In a classification problem, the dimensions of an object can be divided into two types. One dimension records the class type of the object and the rest dimensions are attributes" [231]. In this sense, classification methods aim to synthesize the 'class type' dimension from the other attribute-type dimensions, for all observations present in a dataset.

A typical example of classification concerns datasets where observations represent images, such as photographs stored in large online collections [132], video frames taken from video collections [182], or medical images taken for diagnosis and prognosis purposes [199]. In all such cases, each image, or observation $\mathbf{x}_i \in \mathcal{D}$ has tens up to hundreds of attributes $\mathbf{x}^j \in \mathbb{R}$. These attributes, also called features in image processing and machine learning contexts, describe a wide range of quantities that can be automatically measured on the input images, on the one hand, and arguably capture relevant aspects for further interpretation of the images, on the other hand [245, 49]. Classification, next, can take various forms and serve various purposes. For instance, classification can group all observations $\mathbf{x}_i \in \mathcal{D}$ into a small set of so-called clusters, based on observation similarity, and assign labels to these clusters. Clustering, seen here as a form of unsupervised classification, is described separately in detail in Sec. 2.3.4. This type of application identifies the most salient groups of similar observations, and lets the user the task to assign meanings to the inferred class, or cluster, labels. In contrast, supervised classification starts with a dataset $\mathcal{D}$ where, for each observation $\mathbf{x}_i \in \mathcal{D}$, a class label value is supplied by the user, *e.g.* by manual examination and annotation. Next, given an unlabeled dataset $\mathcal{D}'$, classification aims to associate labels to all entries $\mathbf{x}'_i \in \mathcal{D}'$ based on their similarity, as reflected by non-label attributes, to entries $\mathbf{x}_i \in \mathcal{D}$.

Among the approaches used to classify multidimensional data are k-nearest-neighbors, neural networks [121], decision trees [75], rule-based classifiers [47], and support vector machines (SVMs) [229]. In our context, SVMs are arguably a very interesting type of technique to study, as they involve, by construction, the use of high-dimensional data spaces. Specifically, SVMs transform the n-dimensional space in which the input dataset $\mathcal{D}$ is embedded, into a much higher-dimensional space (and corresponding dataset), so that the separation borders between labeled observations become simple hyperplanes in this high-dimensional space. This way, nonlinear separation borders between the labeled (training) instances in the original n-dimensional space can be achieved.

While the machine learning community has witnessed an explosion of data classification algorithms in the last decade, the practical utilization of such algorithms for understanding multidimensional datasets is still challenged by several aspects. First, it is far from evident, for a given user-dataset-problem combination, which is the best classification algorithm to use. This is caused by the fact that different algorithms may yield different classification results for the same datasets and training sets. Secondly, it is not evident, especially for typical end users who are not machine learning experts, how to fine tune a given classification algorithm to achieve maximal accuracy in a given context [146]. Indeed, many such classification algorithms are either designed to work like black boxes (in which case it is almost impossible for end users to fine tune their behavior), or alternatively expose a number of highly abstract and mathematically involved parameters (in which case typical end users must undergo a steep and costly learning process to understand how to use such parameters). At a high level, we detect the general need, for end users of classification algorithms, of tools and techniques that make the working of such algorithms more transparent and easy to understand.

2.3.4 *Clustering*

Clustering is the process of grouping a set of data observations into multiple groups, also called *i.e.* clusters, so that observations within a cluster have high similarity, while observations belonging to different clusters have low similarity. In this sense, clustering can be seen as an unsupervised classification process (Sec. 2.3.3), which assigns a class (cluster) label to each observation in the input dataset, based on the optimization of inter-observation distances indicated above.

Clustering knows many applications in data analysis and data visualization. One of the best known applications of clustering is *data simplification*: Given a very large dataset $\mathcal{D}$, generate a dataset $\mathcal{D}'$ whose size, measured in number of observations, should be a small user-controlled fraction of the size of $\mathcal{D}$, so that $\mathcal{D}'$ should encode the essence of the phenomenon sampled by $\mathcal{D}$. The construction of $\mathcal{D}'$ from $\mathcal{D}$ can be seen as a subsampling, or alternatively, data clustering problem. An essential ingredient in this process is a distance, or dissimilarity, function $\delta : \mathcal{D} \times \mathcal{D} \to \mathbb{R}^+$ that measures how observations $\mathbf{x}_i \in \mathcal{D}$ differ from each other. Given such a distance function, a very popular clustering method is the bottom-up hierarchical agglomerative technique [53]: Given a dataset $\mathcal{D}$ of multidimensional observations and a distance metric δ as outlined above, all observations $\mathbf{x}_i \in \mathcal{D}$ become nodes in a dendrogram tree. Next, the tree is built bottom-up by grouping the two existing nodes which are most similar with respect to the distance δ. Finally, the resulting tree can be cut at any level to obtain a partition of the original dataset $\mathcal{D}$ into disjoint groups of observations. By choosing different designs for the distance metric δ and tree-cut, a wide range of clustering constraints can be modeled. Such clustering methods have been used in several contexts to make information visualization applications scalable to large datasets [215, 148]. Other popular clustering algorithms used for simplifying large datasets are k means [140], mean shift [44], and self-organizing maps [116].

In all above examples, data clustering takes place in the *observation* space, and its aim is to reduce the number of observations required to describe the structure of a given input dataset $\mathcal{D}$. However, data clustering can also take place in the *dimension* space. In this context, for example, *subspace clustering* methods aim to automatically find and group related dimensions $\mathbf{x}^j$ into clusters. The goal for this operation is to filter out dimensions which are irrelevant for the description of the data at hand, and thereby represent the data with a small number of dimensions. Three examples of subspace clustering techniques are CLIQUE [4], ENCLUS [40], and PROCLUS [2]. CLIQUE enumerates subspaces, and clusters present in these subspaces, following a dimensionality increasing order, pruning subspaces in which no cluster exist. ENCLUS is an exploration method particularly effective for finding data dimensions which are not tightly coupled (correlated). PROCLUS is a k-medoid method that aims to find k potential dimension clusters using a dataset sample and refining the cluster subspaces iteratively. For cases where users want to find a trend in observations in a subset of dimensions, rather than find observatons with similar values, the bicluster model is an option. A bicluster is composed of a set of observations (U) and a subset dimensions (D) such that observations in U have similar trends across dimensions in D. An application example can be found in [41] in the context of the analysis of multidimensional genomic datasets. A data cube aggregation concept is introduced in [83] and it works storing multidimensional aggregated information with a hierarchy for each attribute or dimension, allowing the analysis of data at multiple abstraction levels.

Subspace clustering techniques are related, at a high level, to multidimensional projection techniques, which are discussed separately further in Sec. 2.4.3. Both types of techniques aim to reduce the number of dimensions required to describe the essence of variation captured in the observations of a given dataset. However, important differences exist. Most subspace clustering methods output dimensions which are a *subset* of the original set of dimensions $\mathbf{x}^j$ of the input dataset $\mathcal{D}$. This makes the interpretation of these clustered dimensions relatively easy to do in terms of the original dimensions. In contrast, most multidimensional projection methods output dimensions which are completely *different* from the original dimension-set $\mathbf{x}^j$. For example, the well-known principal component analysis (PCA) technique reduces the n dimensions $\mathbf{x}^j$ to n eigenvectors of the covariance matrix S of the observations (Eqn. 2.4). From these n eigenvectors, a small subset of typically two or three can be next chosen to create a 2D or respectively 3D visual projection of the data. These eigenvectors aggregate the original dimensions $\mathbf{x}^j$, in the sense that they are linear combinations thereof. This makes the interpretation of such synthesized dimensions much harder. This topic is detailed further in Sec. 2.4.3.

2.4 MULTIDIMENSIONAL DATA VISUALIZATION

Besides data analysis methods, such as the ones discussed in Sec. 2.3, multidimensional datasets can be explored also by direct means, by creating suitable visualizations thereof. In this section, we discuss the main types of visualization methods aimed at supporting the direct exploration of multidimensional data.

The added value of multidimensional data visualization is easy to explain in the context of applying the data analysis techniques discussed in Sec. 2.3. At a high level, two main challenges exist to 'pure' data mining applications: (a) In *data exploration* contexts, it is to for such applications to automatically search for patterns when one does not know how to define such patterns (one does not know what one searches for); (b) For applications involving *casual end-users*, presenting the results of data mining in text or tabular forms is far from being intuitive and effective.

The key role and added-value of visualization is to take advantage of the human cognitive skills by creating visual depictions of data from which users can detect data-related *patterns* with ease. Such patterns involve, but are not limited to, correlations of variables; clusters and outliers defined in terms of observations; and trends defined in terms of variable changes over subsets of observations. As mentioned earlier in Sec. 2.2.2, the main added-value of visualization, as opposed to pure data analysis methods, is that many kinds of patterns are relatively easy to describe in an informal way, and relatively easy to visually detect, but are (very) hard to describe and detect in an automated manner. Finding and reasoning about the meaning of these patterns, by visual means, connects to the two main aims of data visualization: *confirm the known* (find evidence in data that matches an established model of the underlying phenomenon); and *discover the unknown* (find new, unexpected, patterns in data that lead to new hypotheses about the underlying phenomenon).

In this context, a recent area called *visual data mining* [240] is calling attention in the computer science literature. Visual data mining fuses classical data mining techniques (Sec. 2.3) with classical data visualization techniques [198, 147, 217], by effectively exploiting (1) the computational scalability of data mining techniques and their ability to perform complex exact queries, and (2) the ability of the human visual system to quickly detect complex, fuzzily-specified, patterns in images, and extract information from such patterns [233]. This way, data mining and data visualization jointly complement each others' strengths. In the visual data mining process, Ankerst [7] has identified three different ways in which data visualization and data mining can be integrated together: (1) visualization methods are applied before or independent of data mining technique; (2) the results are obtained with data mining techniques first, and visualization is used next to provide support to do the data knowledge extraction; or (3) visualization and data mining are tightly integrated in an iterative and interactive process that combines activities of types (1) and (2) above.

To understand how visualization can address the above-mentioned data exploration tasks in the context of visual data mining applications, we next discuss the most known methods used for visual exploration of multidimensional data. We group these into four classes: axis-based methods, space-filling approaches, multidimensional projections, and other approaches. We note that alternative taxonomies for visualization methods for multivariate data exist, *e.g.* those proposed by Keim and Krieger [114, 112] and further refined by Chan [36]. We use here a different classification so as to better emphasize the special class of multidimensional projection techniques, which will play a salient role in the remainder of this thesis.

2.4.1 *Axis-Based Methods*

Axis-based methods represent the values of all attributes $\mathbf{x}^j$ of a dimension j by mapping them along an axis, much like in when plotting 1D scalar values along a line. Different axes are used to represent the different n dimensions of a multidimensional dataset. By suitably arranging these axes, such methods support reasoning about the entire set of m observations and n variables.

Scatterplots are one of the best known, and most frequently used, methods used to visualize $n = 2$ dimensional or $n = 3$ dimensional datasets. Their design directly follows the classical design of function graphs: Each observation $\mathbf{x}_i$ is mapped to a point placed in 2D or 3D space, based on its attribute values. Scatterplots support a range of assessments about the underlying dataset. For example, attribute ranges densely populated by observations appear as dense 'spots' in the resulting 2D or 3D point cloud. To reduce visual clutter and, in the same time, help assessing the observation density and focus the user's attention on densely populated regions, simple techniques such as transparency and alpha blending of points can be used. The attribute values corresponding to these ranges can be easily inferred by looking at the respective coordinate values in the scatterplot. Clusters of similar observations can be easily perceived, since 2D or 3D distance (in the attribute space) is mapped one-to-one to 2D or 3D distance in the resulting plot. Outlier observations can be easily spotted as points in the scatterplot which are far away from densely-populated regions. Finally, direct or inverse correlations between the $n \in \{2, 3\}$ variables can be relatively easily spotted by searching for point-spreads along straight lines (in 2D) or planes (in 3D) in the plot. Conversely, the lack of correlation between variables can be spotted by looking for 2D or 3D 'clumps' of points that do not follow such linear patterns. However, scatterplots cannot, by construction, directly visualize datasets having more than $m = 3$ dimensions. Also, even for the case $m = 3$, understanding 3D scatterplots is considerably more challenging as compared to understanding 2D scatterplots, due to inherent problems related to visualizing 3D point clouds, such as occlusion, difficulty to choose a good viewpoint, parallax effects, and the difficulty of estimating 3D point cloud density.

To extend the dimensional scalability of scatterplots, additional variables can be encoded into per-point visual attributes, such as color, shape, or size. This gives good results once one knows *a priori* which dimensions to assign to this small set of visual variables. For high-dimensional datasets (high n values), this is a complex problem, which in the limit forces users to cycle through the entire set of n variables to map them to *e.g.* color in order to detect potential patterns. Separately, encoding variables into shape or size imposes constraints on the minimal point size, which in turn can easily lead to undesired occlusion and visual clutter for datasets having many observations.

Scatterplot matrices extend the idea of 2D scatterplots by constructing a so-called *small multiple* layout [222], also known under the name of *trellis plot* [13], and discussed even earlier under the generic name of *collections* [19]. In detail, given an n-dimensional dataset, a matrix of $n \times n$ 2D scatterplots is constructed, where scatterplot (i, j) shows the relationship between variables $\mathbf{x}^i$ and $\mathbf{x}^j$ in the original dataset. The result is called a scatterplot matrix, or SPLOM. The advantage of SPLOMs is that they show, in detail, all relationships between pairs of variables in the original dataset (Fig. 2.1). However, the scalability of SPLOMs is obviously limited by the number of dimensions n of the input dataset. For typical output devices (*e.g.* computer screens) and tasks, SPLOMs scale up to roughly $n \in \{10, \ldots, 20\}$ dimensions. Handling multidimensional datasets having hundreds or even thousands of dimensions is not possible. A second problem of SPLOMs relates to the fact that they only show relations betwen *pairs* of variables. In multidimensional datasets, however phenomena of interest may be describable only when we analyze larger numbers of variables together. This is not directly supported by SPLOMs. This is recognized by Hand *et al.* that call SPLOMs 'multiple bivariate infovis techniques', in contrast to multivariate techniques [84]. A third problem of SPLOMs is the fact that they adopt a dimension-centric view: While it is relatively easy to reason about *dimensions* in a SPLOM, as these appear explicitly as rows and columns of the matrix, reasoning about *observations* is harder, as a single observation appears as n^2 different points, one point per 2D scatterplot in the SPLOM. While such issues can be partially corrected by interaction techniques such as linked views, *e.g.* by brushing a scatterplot and highlighting brushed observations in all other scatterplots, it is still hard to reason about observations in SPLOMs.

To improve the idea of exploring sets of 2D scatterplots, Elmqvist *et al.* proposed the 'rolling the dice' metaphor. In this set-up, users can interactively and continuously morph the view on data between consecutive 2D scatterplots of chosen variables i and j of their multidimensional dataset. By using animation to linearly interpolate the positions of observations in consecutive scatterplots, users can thereby analyze larger sets of variables than pairs [64]. A related interaction mechanism was proposed by Hurter *et al.* for exploring multidimensional data such as air flight information [92] and 3D multivariate fields [97]. Here, the user can actively control the transition between two scatterplots, by effectively 'morphing' one into the other. By playing such transitions a few times back and forth, one can thus visually link related structures in different spaces. Separately, by stopping the transition at any desired intermediate stage, data-related structures which are not apparent in any of the end views can be spotted.

On the positive side, such techniques significantly reduce the amount of necessary screen space for visualizing the entire dataset, by factoring out additional dimensions to the animation and interaction side. On the negative side, such techniques strongly rely on the visual memory of their users in terms of being able to remember (and correlate) information and insights shown at different moments in time, and being visible only if the suitable interactions have been chosen.

Scagnostics aim to address some of the scalability issues of SPLOMs. These techniques propose a set of measures that help to identify the most interesting scatterplots in a SPLOM, to next focus the user's investigation on these [223]. Wilkinson *et al.* refined this idea considering the five aspects of scattered points (outliers, shape, trend, density and coherence), creating nine quality measures derived from geometric graphic features [239]. Later, Lehmann *et al.* proposed a set of different measures to find clusters of similar relevance in scatterplots [125]. These clusters can be visualized by color coding and dimension reordering in a SPLOM global view. One drawback of this approach is that different clusters of large relevance could be caused by different initial orders of dimensions, something that is not evident in the proposed views. Essentially, scagnostics combine data analysis (in terms of automatically ranking 'interesting' scatterplots or parts thereof) and data visualization (in terms of directly showing the highly-ranked plots) in a single approach. As such, their main limitation relates to the inherent limitation of data analysis methods in terms of being able to automatically find interesting patterns: If such patterns are found, then such approaches are obviously effective. However, if interesting patterns are missed (which can

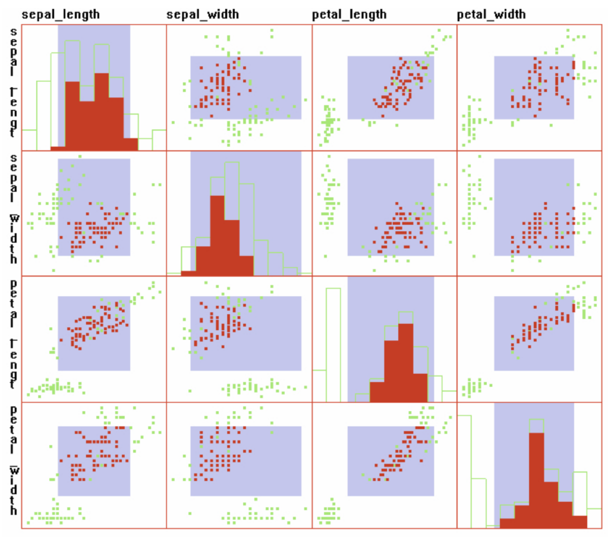


Figure 2.1: Example of a SPLOM (*iris* dataset) [233].

easily happen when we cannot fully quantify what 'interesting' means), then such approaches may fail to convey important insights in the data.

Parallel coordinate plots, or PCPs, can be seen, in some sense, as the 1D equivalent of SPLOMs. They consist of a set of n parallel, and typically vertically drawn, axes, one per data dimension $\mathbf{x}^j$ (Fig. 2.2). Each value x_i^j maps to a point along the j^{th} axis. Each observation $\mathbf{x}_i$ maps to a polyline connecting all points x_i^j, $1 \leqslant j \leqslant n$ [99]. PCPs support several tasks, as follows. Distributions of values $\mathbf{x}^j$ around a given variable j can be easily spotted by looking at the point density along PCP axis j. Brushing and selection helps narrowing down the analysis to a specific set of observations, by *e.g.* selecting a value range on an axis; outliers can be easily found by looking for polylines which are far away from the main trend (Fig. 2.2 a). To further enhance this understanding, explicit histograms can be added, in terms of bar charts, along the axes; and axis directions (top to bottom *vs* botto to top) can be swapped to minimize undesired crossings of polylines (Fig. 2.2 b). Similarly, outlier values or ranges along an axis j can be spotted by looking at concentrated sets of points along that axis which are well separated by other points (along the same axis) by large amounts of white space. More interestingly, correlations between variables plotted to adjacent axes i and j in the PCP can be spotted by locating dense sets of close, nearly parallel lines that connect axes i and j. Inverse correlations (between such axes) can be spotted by locating an 'X' like pattern formed by polyline segments connecting those axes. Lack of correlation (between such axes) can be spotted by locating a thick band of lines crossing each other at arbitrary angles between the respective axes.

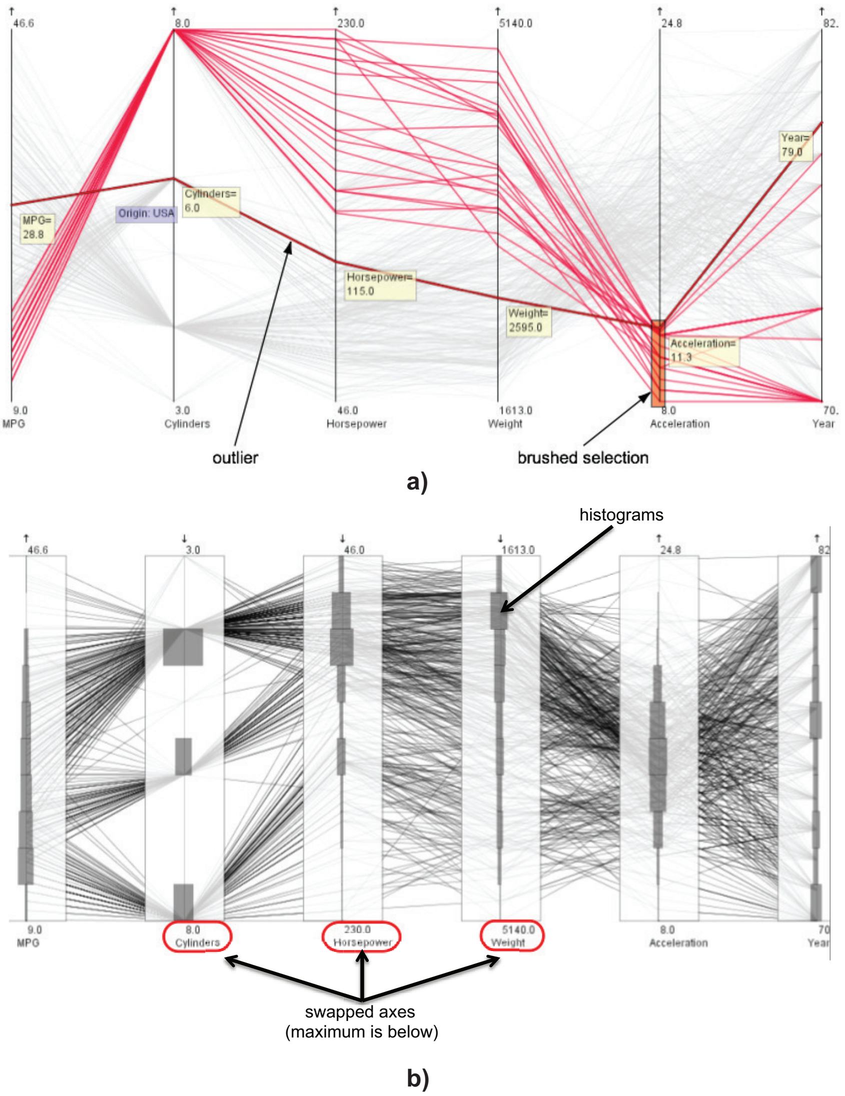
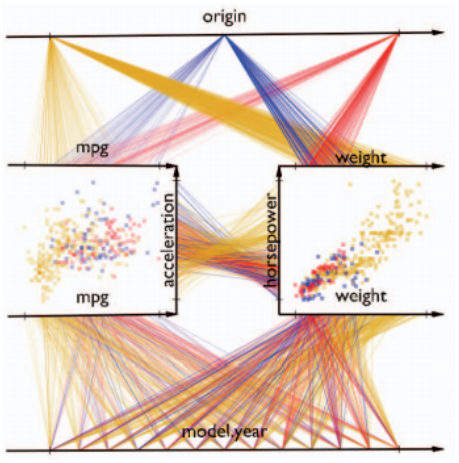


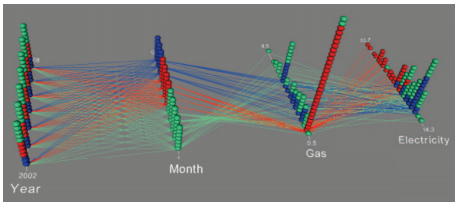
Figure 2.2: Example PCP visualizations using the *parvis* toolkit [124].

Sets of similar observations can be spotted by looking at end-to-end polylines which are nearly parallel and close to each other. To reduce occlusion and clutter, transparency and alpha blending can be applied to polylines, much as done for traditional 2D scatteplots (see above).

However, PCPs have also a number of challenges. Axis ordering is arguably the main one: Unless axes are ordered (along the horizontal screen dimension) in a suitable way, it can be very hard to compare relevant variables, and thus to spot relevant patterns such as correlations. The issue here is that one does not know the optimal sorting, since this depends on insights which are only known (and visible) one such an ideal sorting has been achieved. As there are $n!$ such sortings possible for n variables, the space to explore is clearly large. Secondly, PCPs are limited in terms of scalability by the available horizontal screen space. For typical displays, this allows one to explore about 10..20 variables, but not more. To avoid the visual clutter related to classical PCPs, Dang *et al.* proposed a PCP where overplotting is handled by stacking overlapping elements in 3D [50] . Dot 'towers' along each PCP axis highlight major differences in frequency that are not evident in the traditional PCP. A different way to overcome the layout limitations of classical PCPs that use parallel axes is presented by Claessen *et al.* in their *FLINAPlot* design [43]. This approach, on the one hand, combines the strengths of PCPs and scatterplots in a single view, and on the other hand allows users to interactively arrange the PCP axes in a wide range of configurations, by drawing them explicitly on screen. While this approach can, in principle, eliminate some of the limitations of the classical PCP, it also requires a considerable amount of interaction effort from the end user. Additionally, exposing design freedom to end users can be risky, since one can (easily) create PCPs that do not convey the required information.



(a) *FLINAPlots* for the cars dataset: origin axis highlighted (yellow: USA; blue: Europe; red: Japan) [43].

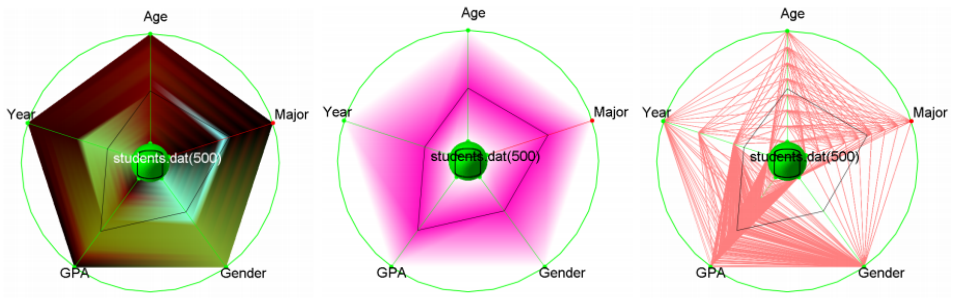


(b) View of the *Utility* data set using parallel coordinate dot plot [50].

Figure 2.3: Examples of PCP visualizations.

Radial layouts adapt the idea of PCPs by proposing a different layout of the variable axes. Essentially, axes are arranged in a radial layout, rather than in a sequential layout (*e.g.* along the screen horizontal axis). The main added-value of this design is that many more axes can be fitted in limited screen space. Additionally, the resulting plot is guaranteed to have a 1 to 1 aspect ration, something that cannot be done by the traditional PCP by construction [90]. Star Coordinates use a similar approach to the RadViz design [90], where the variable axes are computed as unit basis vectors of an affine projection [108, 109]. A similar concept is offered by Star Plots [35]. However, this design can introduce undesired distortions due to user manipulations. Similar radial layouts are proposed by DataRoses and DataMeadows [63] (see Fig. 2.4).

While radial layouts achieve better aspect ratios than traditional PCPs, they have a number of limitations. First, the inherent problem of optimal axis ordering present in traditional PCPs is apparent here too. Secondly, radial layouts do not preserve distances in the same way that Cartesian PCPs do – polylines closer to the origin appear, overall, shorter, less important, and more thus similar than polylines far away from the origin. This creates an undesired bias towards better seeing large-value observations. Finally, correlations (direct or inverse) map to relatively simple patterns in traditional PCPs. In radial layouts, such correlations appear rotated to various angles, depending on the position of the respective axes in the plot. This makes them harder to spot, analyze, and compare.



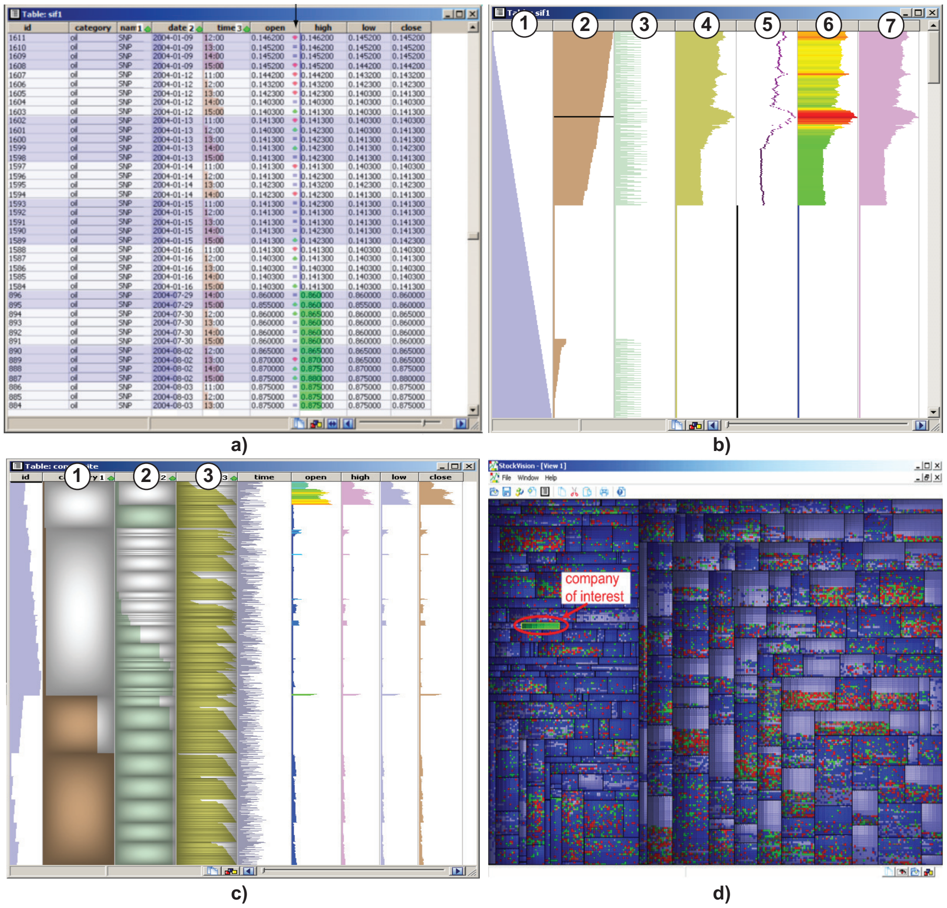
(a) Color histogram mode (high brightness equals high density)

(b) Opacity bands mode

(c) Parallel coordinate mode

Figure 2.4: Examples of radial layout: *DataMeadow* [63]. Sample DataRose visualization for a university student database of a computer science department.

Table lenses take a more traditional, but remarkably powerful, approach to visualizing multidimensional data. They regard a multidimensional dataset of m n-dimensional observations $\mathbf{x}_i = (x_i^1, \ldots, x_i^n)$, $1 \leqslant i \leqslant m$ as a table where each observation $\mathbf{x}_i$ is a row and each dimension $\mathbf{x}^j$ is a column, respectively. For small numbers m of observations, the data is drawn as a classical Cartesian table, where each cell conveys the value x_i^j in textual form, optionally overlaid with a semi-transparent bar charts, scaled and colored to reflect the data value (Fig. 2.5 a). Scrolling the table along observations i allows seeing variations (trends) in individual variables $\mathbf{x}^j$ by spotting the changes of their corresponding bar charts. To visualize large tables, one can zoom out the view and reduce each table row to a line of pixels (Fig. 2.5 b). In this mode, cell text is not drawn (since too small to be readable), and the entire view reverts to a set of vertical bar charts [184]. Additionally, sorting the table rows along the value of one variable $\mathbf{x}^j$ directly shows the (inverse) correlation of that variable with other variables $\mathbf{x}^{k \neq j}$ in the form of the variables' bar graphs. For example, in Fig. 2.5 b, the table is sorted on increasing values of column 1. This view immediately shows how columns 1 and 2 are inversely correlated; that columns 4 up to 7 are directly correlated; and that column 3 is not correlated with any other column.



id	category	nam 1	date 2	time 3	open	high	low	close
1611	oil	SNP	2004-01-09	12:00	0.146200	0.146200	0.145200	0.145200
1610	oil	SNP	2004-01-09	13:00	0.145200	0.146200	0.145200	0.145200
1609	oil	SNP	2004-01-09	14:00	0.145200	0.145200	0.145200	0.145200
1608	oil	SNP	2004-01-09	15:00	0.145200	0.145200	0.144200	0.144200
1607	oil	SNP	2004-01-12	11:00	0.144200	0.144200	0.143200	0.143200
1606	oil	SNP	2004-01-12	12:00	0.143200	0.143200	0.142300	0.142300
1605	oil	SNP	2004-01-12	13:00	0.142300	0.142300	0.140300	0.141300
1604	oil	SNP	2004-01-12	14:00	0.140300	0.140300	0.140300	0.140300
1603	oil	SNP	2004-01-12	15:00	0.140300	0.141300	0.140300	0.141300
1602	oil	SNP	2004-01-13	11:00	0.141300	0.141300	0.140300	0.140300
1601	oil	SNP	2004-01-13	12:00	0.140300	0.142300	0.140300	0.141300
1600	oil	SNP	2004-01-13	13:00	0.141300	0.142300	0.141300	0.141300
1599	oil	SNP	2004-01-13	14:00	0.141300	0.142300	0.141300	0.142300
1598	oil	SNP	2004-01-13	15:00	0.141300	0.142300	0.141300	0.141300
1597	oil	SNP	2004-01-14	11:00	0.141300	0.141300	0.140300	0.140300
1596	oil	SNP	2004-01-14	12:00	0.141300	0.142300	0.141300	0.141300
1595	oil	SNP	2004-01-14	13:00	0.142300	0.143200	0.142300	0.142300
1594	oil	SNP	2004-01-14	14:00	0.142300	0.142300	0.141300	0.141300
1593	oil	SNP	2004-01-15	11:00	0.141300	0.142300	0.141300	0.141300
1592	oil	SNP	2004-01-15	12:00	0.141300	0.141300	0.141300	0.141300
1591	oil	SNP	2004-01-15	13:00	0.141300	0.141300	0.141300	0.141300
1590	oil	SNP	2004-01-15	14:00	0.141300	0.142300	0.141300	0.141300
1589	oil	SNP	2004-01-15	15:00	0.141300	0.142300	0.141300	0.142300
1588	oil	SNP	2004-01-16	11:00	0.141300	0.141300	0.140300	0.140300
1587	oil	SNP	2004-01-16	12:00	0.140300	0.141300	0.140300	0.141300
1586	oil	SNP	2004-01-16	13:00	0.140300	0.140300	0.140300	0.140300
1585	oil	SNP	2004-01-16	14:00	0.140300	0.141300	0.140300	0.140300
1584	oil	SNP	2004-01-16	15:00	0.140300	0.141300	0.140300	0.141300
896	oil	SNP	2004-07-29	14:00	0.860000	0.860000	0.860000	0.860000
895	oil	SNP	2004-07-29	15:00	0.855000	0.860000	0.855000	0.860000
894	oil	SNP	2004-07-30	12:00	0.860000	0.865000	0.860000	0.865000
893	oil	SNP	2004-07-30	13:00	0.860000	0.860000	0.860000	0.860000
892	oil	SNP	2004-07-30	14:00	0.860000	0.860000	0.860000	0.860000
891	oil	SNP	2004-07-30	15:00	0.860000	0.860000	0.860000	0.860000
890	oil	SNP	2004-08-02	12:00	0.865000	0.865000	0.865000	0.865000
889	oil	SNP	2004-08-02	13:00	0.870000	0.870000	0.865000	0.865000
888	oil	SNP	2004-08-02	14:00	0.870000	0.875000	0.870000	0.875000
887	oil	SNP	2004-08-02	15:00	0.875000	0.880000	0.875000	0.880000
886	oil	SNP	2004-08-03	11:00	0.875000	0.875000	0.875000	0.875000
885	oil	SNP	2004-08-03	12:00	0.875000	0.875000	0.875000	0.875000
884	oil	SNP	2004-08-03	13:00	0.875000	0.875000	0.875000	0.875000

Figure 2.5: Table lens example. (a) Zoomed in table with text and bar charts. (b) Zoomed out table. (c) Table sorted and grouped on first three columns. (d) Table hierarchy visualized with a treemap. Images generated with the TableVision tool [215].

The table lens technique allows a quite significant scalability: The number of columns n displayed simultaneously can easily reach several tens on a typical computer screen. The number of rows displayed simultaneously can reach a few thousands (depending on the screen vertical resolution). If aggregation techniques for the types of attributes present in the table are available (*e.g.* averaging, maximum, minimum, or similar), the table lens can, in principle, visualize an unbounded number of observations.

Several enhancements have been proposed for table lenses. Contiguous rows having the same values for an attribute $\mathbf{x}^j$ can be emphasized by overlaying them by shaded cushions [228], to emphasize attribute value distributions [215]. Separately, multiple sort-and-group operations executed on different attributes j create on-the fly hierarchies of the data values, which can be next visualized either by the basic table lens, or alternatively by treemap techniques [215]. This essentially allows users to interactively explore the data by executing chains of operations equivalent to the well-known SQL commands 'ORDER BY' and 'SORT BY'. Figure 2.5 c shows a data table with rows sorted and grouped by values of columns 1, 2, and 3, with same-value cells emphasized by shaded cushions. The three-level data hierarchy thus created is next visualized using squarified cushion treemaps [228, 30] (Fig. 2.5 d).

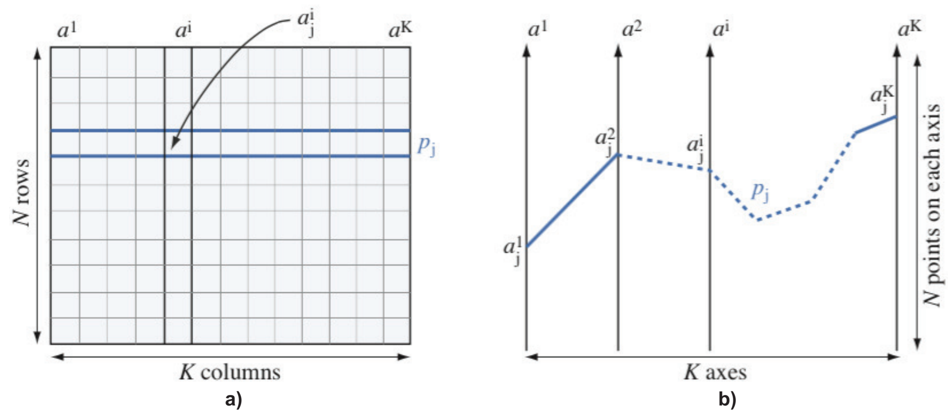


Figure 2.6: Comparison of table (a) *vs* (b) PCP visualization layouts. Image taken from [217].

However powerful, table lenses also have limitations. First, the proposed data visual layout is constrained by the fact that an observation (table row) is always a horizontal line (Fig. 2.6 a). This does not support tasks such as finding groups of observations which are similar – a task well supported by *e.g.* PCPs (Fig. 2.6 b). Secondly, comparing different columns can be hard if these columns are placed far away from each other, in the horizontal table layout – a problem basically identical to the axis ordering in PCPs. Thirdly, observations are less prominent (in terms of visual encoding) than in PCPs. Indeed, in table lenses an observation is a set of horizontally laid out bars, while in PCPs an observation is a polyline (which has a more salient visual identity). On the positive side, table lenses inherit the ease of interpretation – arguably a large range of users understand what a data table is, whereas significantly fewer users are comfortable with the more abstract depiction of data proposed by PCPs or SPLOMs.

2.4.2 *Space Filing Approaches*

Multidimensional datasets create significant scalability challenges for visualization techniques, both in terms of number m of observations and number n of dimensions. The visualization techniques discussed so far, albeit effective, do not use all the available screen space to the maximum. Indeed, they all reserve a certain amount of white space (screen pixels not used to encode information) in order to separate the drawn information and make it readable. In the same time, they address the scalability issue by techniques such as overdraw (scatterplots, SPLOMs, PCPs) or data aggregation (table lenses). This makes it hard, or even impossible, to reason about individual observations.

To address these issues, *space filling* approaches have been proposed. Also known under the name of pixel-based techniques or dense pixel displays, such approaches aim to encode a maximal amount of information on the available screen space, and in the same time avoid (by construction) overlapping data elements. The main challenge of such designs is to find suitable mappings between the $m \times n$ data values to map into the 2D screen space, so that data patterns of interest become easily visible. One such design uses the small multiple metaphor to create a separate visualization for each of the n dimensions. Within each such visualization, the m observation values are laid out in a space-filling manner, and visualized by color coding. One way to construct such per-dimension layouts is to order the pixels along a 1D space-filling curve structure and map this in a 2D space (2D space-filling curve), based on the type of query one wants the visualization to support. Figure 2.7 illustrates this for the visualization of 1000 8-dimensional observations with the *VisDB* tool [113]. Here, the observations which are most relevant to the query to execute on the data are mapped to the centers of the per-dimension views; the data then spirals outwards as it becomes less relevant to the query. Keim *et al.* created a similar visual representation based on grid subimages in which users could reorder the dimensions, helping to reveal possible correlations between dimensions. Ankerst *et al.* use the same idea in their *circle segments* design, and offer similar user flexibility in terms of data reordering, but proposed a radial layout of dimensions instead of a grid (Cartesian) one [8]. Besides spirals, other types of space-filling curves can be used. Curves that keep points close in the order (along the curve) also in the 2D screen space, such as Hilbert, Peano-Hilbert, Morton and H-curves, are examples hereof. Following the same idea, Wattenberg introduced the *jigsaw map*, which maps data points to pixels and uses discrete space-filling curves in order to fill a 2D plane in a better way than classical treemap layouts, *i.e.* ensuring a better aspect-ratio of the leaf nodes, without distortion.

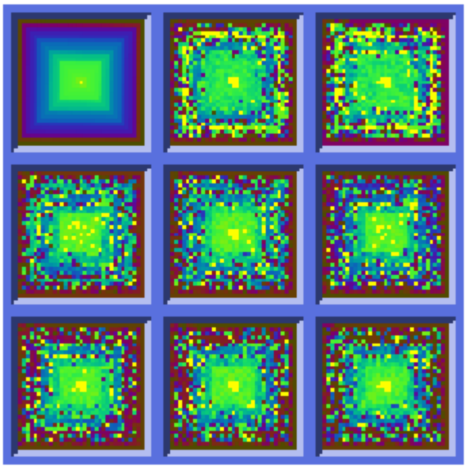


Figure 2.7: VisDB [113]. Color coding ranges from yellow for those data items that better satisfy a posed query to green, blue, red, and almost black for those further away from it.

2.4.3 *Multidimensional Projections*

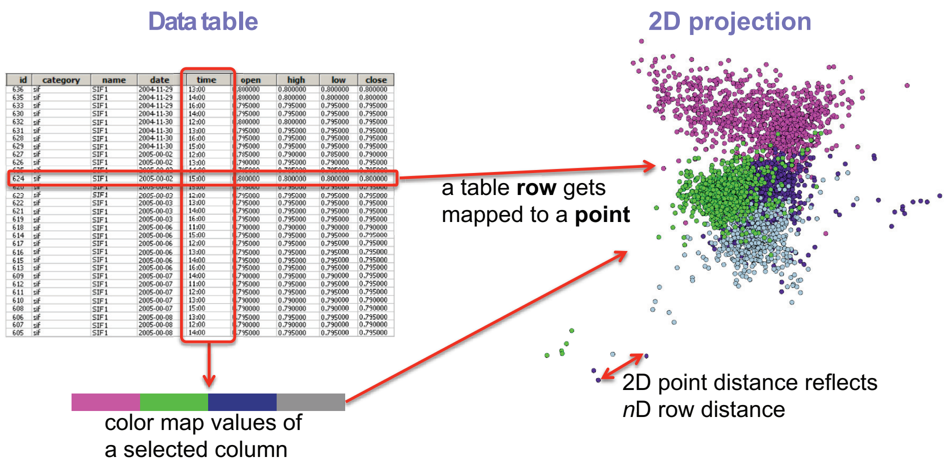
The visualization methods discussed so far take different approaches to (1) making high-dimensional patterns visible, (2) explicitly showing observations and attributes, and (3) ensuring scalability in terms of the maximal number of dimensions and observations being shown simultaneously. However, as our discussion has shown, no single method is able to fully satisfy (1), (2), and (3) equally well.

Dimensionality reduction techniques, also called multidimensional projections, or more simply projections, take a different approach than all earlier methods. Intuitively put, they recognize (and aim to address) two important limitations of all earlier methods:

- *visual scalability:* Large and high-dimensional datasets having many observations m and dimensions n, it may be simply impossible to map all information present in the data to a single screen, without occlusion. Indeed, a typical PC screen having roughly $P = 2$ million pixels can show at most $P = m \times n$ individual data items. However, current multidimensional datasets obtained *e.g.* from text mining [91, 18] can easily generate tens of thousands of observations having each thousands of dimensions;

- *visual conciseness:* All methods discussed so far map can be thought as being more *attribute-centric* than *observation-centric*. That is, their visual design is geared towards easily understanding attributes and their values, rather than understanding observations. Indeed, an observation is mapped to relatively complex visual shapes – horizontally aligned bars (table lens), polylines (PCPs and their variations), a set of $n \times n$ points located in separate 2D scatterplots (SPLOMs), and a set of n colored dots located in separate dense-pixel displays (space-filling approaches). This makes it relatively hard for users to see an observation 'at a glance', as its visual encoding becomes increasingly more complex with the number of dimensions n. Separately, increasingly complex shapes affect visual scalability, due to increased potential for clutter and overdraw.

At a high level, projection methods essentially recognize that (a) it is very hard, if not impossible, to show all information involving the m observations and their n attributes in a multidimensional dataset; and (b) that visually encoding *observations* by simple to understand elements may be, in many cases, favorable. In this context, multidimensional projections can be defined, in the terminology of Tejada *et al.*, as follows [214]: Given a set $X = \{\mathbf{x}_i\} \subset \mathbb{R}^n$ of observations, and a so-called 'criterion of proximity' or similarity function $\delta : \mathbb{R}^n \rightarrow \mathbb{R}_+$ between items in $\mathbb{R}^n$; and given a low-dimensional space $Y \subset \mathbb{R}^p$, where $p << n$ (typically, $p \in \{1, 2, 3\}$), and a corresponding similarity criterion or similarity function $d : \mathbb{R}^p \rightarrow \mathbb{R}_+$; with this notations, a *multidimensional projection* is a mapping $f : X \rightarrow Y$, so that $|\delta(\mathbf{x}_i, \mathbf{x}_j) - d(f(f(\mathbf{x}_i), f(\mathbf{x}_j))|$ is as close as possible to zero, $\forall \mathbf{x}_i, \mathbf{x}_j \in X$. In other words, multidimensional projection methods aim at mapping instances from a high dimensional space X to a low dimensional space Y (also called the *embedding* space) by preserving inter-observation distances as much as possible.



id	category	name	date	time	open	high	low	close
636	sif	SIF1	2004-11-29	13:00	.800000	0.800000	0.800000	0.800000
635	sif	SIF1	2004-11-29	14:00	.800000	0.800000	0.800000	0.800000
633	sif	SIF1	2004-11-29	16:00	.795000	0.795000	0.795000	0.795000
630	sif	SIF1	2004-11-30	14:00	.795000	0.795000	0.795000	0.795000
632	sif	SIF1	2004-11-30	12:00	.800000	0.800000	0.795000	0.795000
631	sif	SIF1	2004-11-30	13:00	.795000	0.795000	0.795000	0.795000
628	sif	SIF1	2004-11-30	16:00	.795000	0.795000	0.795000	0.795000
629	sif	SIF1	2004-11-30	15:00	.795000	0.795000	0.795000	0.795000
627	sif	SIF1	2005-00-02	12:00	.785000	0.790000	0.785000	0.790000
626	sif	SIF1	2005-00-02	13:00	.790000	0.795000	0.790000	0.795000
624	sif	SIF1	2005-00-02	15:00	.800000	0.800000	0.800000	0.800000
623	sif	SIF1	2005-00-03	12:00	.795000	0.795000	0.795000	0.795000
622	sif	SIF1	2005-00-03	13:00	.795000	0.795000	0.795000	0.795000
621	sif	SIF1	2005-00-03	14:00	.795000	0.795000	0.795000	0.795000
619	sif	SIF1	2005-00-03	16:00	.795000	0.795000	0.795000	0.795000
618	sif	SIF1	2005-00-06	11:00	.790000	0.790000	0.790000	0.790000
614	sif	SIF1	2005-00-06	15:00	.795000	0.795000	0.795000	0.795000
617	sif	SIF1	2005-00-06	12:00	.795000	0.795000	0.795000	0.795000
616	sif	SIF1	2005-00-06	13:00	.795000	0.795000	0.795000	0.795000
615	sif	SIF1	2005-00-06	14:00	.795000	0.795000	0.795000	0.795000
613	sif	SIF1	2005-00-06	16:00	.795000	0.795000	0.795000	0.795000
609	sif	SIF1	2005-00-07	14:00	.790000	0.795000	0.790000	0.795000
612	sif	SIF1	2005-00-07	11:00	.795000	0.795000	0.795000	0.795000
611	sif	SIF1	2005-00-07	12:00	.795000	0.795000	0.795000	0.795000
610	sif	SIF1	2005-00-07	13:00	.790000	0.790000	0.790000	0.790000
608	sif	SIF1	2005-00-07	15:00	.790000	0.790000	0.790000	0.790000
606	sif	SIF1	2005-00-08	13:00	.795000	0.795000	0.795000	0.795000
607	sif	SIF1	2005-00-08	12:00	.790000	0.790000	0.790000	0.790000
605	sif	SIF1	2005-00-08	14:00	.795000	0.795000	0.795000	0.795000

Figure 2.8: Conceptual representation of multidimensional projections for reducing a data table to a 2D scatterplot.

The core added-value of projections can be explained as follows (see also Fig. 2.8). Consider a data table of m rows (observations) and n columns (dimensions). Drawing such a data table requires, using *e.g.* the table lens technique, drawing $m \times n$ data values. Moreover, observations are hard to grasp in the resulting visualization, as these correspond to data-table rows. If we projected this data table in a 2D space, using the projection technique, we would obtain a scatterplot having m points, one per observation. In this plot, inter-point distances attempt to reflect similarities between the original n-dimensional points. This would (a) allow us to detect groups of similar observations as clusters of close points in 2D; outlier observations as 2D points being far away from other groups of 2D points; and, possibly, trends and correlations as specific visual structures in 2D. Additionally, (b) this type of projection plot would map every n-dimensional observation to a *simple* shape: a 2D point in the projection. This would arguably make the projection plot compact and scalable to large amounts of observations *and* large amounts of dimensions. Additionally, coloring projected observations (points) by *e.g.* the value of one attribute from the n-dimensional space could explain the reason why points are similar. Finally, by controlling or constraining the position of a (small) subset of projected observations, and recalculating the projection based on these samples, projections allow users to shape the resulting visualization. This offers effective ways to organize or arrange the projected data in ways which address several tasks, *e.g.* make distances in the projection space reflect some application-dependent notion of similarity, or place projected points at specific locations to obtain an image which is easier to interpret.

Multidimensional projections can be further classified according to different criteria. A first such criterion regards the *information* being used to construct the projection. In this sense, two projection method classes can be found:

- *multidimensional scaling methods:* These methods use, as input to construct the projection function $f : X \to Y$, only the inter-point distances between all pairs of observations $(\mathbf{x}_i, \mathbf{x}_j) \subset X$. The actual dimensions, or attributes x_i^j, of the points $\mathbf{x}_i$, are not known. These methods are also known under the name of multidimensional scaling (MDS) methods. The main advantage of these methods is that they only need the n-dimensional distances, or similarities, between observations, and are otherwise agnostic of the dimensionality n of the original space, as well as of the explicit actual coordinates (or attributes) of the points $\mathbf{x}_i$ in this space. In practical terms, this means that, as long as we are able to provide a distance matrix $D = (\delta(\mathbf{x}_i, \mathbf{x}_j)_{1 \leqslant i,j \leqslant m})$, these methods can generate a low-dimensional projection for us. The main disadvantage of these methods is that computing and storing such a full distance matrix D is quadratic in the number of observations m;

- *projection methods:* These methods use, as input to construct the projection function $f : X \to Y$, all the actual coordinates x_i^j of all the m observations in the input n-dimensional space. Traditionally, these methods are known under the name of projections, which reflects their mapping of n-dimensional points to p-dimensional points. Advantages of these methods, in contrast to MDS methods, are the direct access to the 'raw' n-dimensional observation data, which can be deemed to be more exact, or close to the actual information to visualize, than the access to the derived distance matrix D. Additionally, such methods require as input only an $m \times m$ data matrix, which is typically (much) smaller than the $m \times m$ distance matrix D (and also does not explicitly require computing d-D distances). As typically $n << m$, the storage requirements of projection methods are much smaller than the equivalent storage requirements of MDS methods. On the negative side, projections require explicit access to the attributes of the n-dimensional points, which make such methods more selective, and less generic, than MDS methods.

A second classification of dimensionality reduction methods is based on their algorithmic functioning. in this respect, we can talk about *global* and *local* methods, as follows.

Global methods map data from a high-dimensional space to a low dimensional (visual) space using a single transformation. That is, the function $f : X \rightarrow Y$ does not depend on the local distribution of the n-dimensional observations $\mathbf{x}_i$, but only on *aggregated* properties of the entire set X of observations.Techniques that use spectral decomposition are good examples of global methods. These compute the low-dimensional embedding coordinates for each observation from eigenvectors of a transformation applied to the dissimilarity matrix $D = (\delta(\mathbf{x}_i, \mathbf{x}_j)_{1 \leqslant i,j \leqslant m})$ [220]. Alternatively, global methods can directly use the observations, rather than the distance matrix D. A very simple example of this approach is to perform principal component analysis (PCA) of the covariance matrix of all observations $\mathbf{x}_i$, next select the k largest eigenvectors $\mathbf{e}_j$, $1 \leqslant j \leqslant k$, and $\mathbf{e}_2$, and finally compute the k-dimensional embedding of the data by projecting $\mathbf{x}_i$ on the hyperplane defined by $\mathbf{e}_j$. If $k = 2$, then this method effectively projects the observations on the 2D plane in which the data have the largest spread.

In order to decrease the high computation costs (typically $O(m^3)$, where m is the number of instances to be projected) associated with the eigendecomposition, Roweis and Saul [186] presented an $O(m^2)$ algorithm that combines local fitting and global linear mapping. Sample instances were exploited by Brandes and Pich [24] and De Silva and Tenenbaum [54] achieve an $O(k^3 + km)$ algorithm, where k is the number of samples used to build the mapping. Subsets of samples have been further exploited by Faloutsos and Lin in their FastMap algorithm [69] so as to obtain an $O(m)$ dimensionality reduction scheme. Tenenbaum et al. [218] proposed an $O(m)$ algorithm using a geometric framework. Multiscale matrix representations were employed by Belkin and Niyogii [15] and Koren *et al.* [119] to reduce the computational effort.

Kruskal [122] was, to our knowledge, the first to propose the so-called nonlinear-optimization-based techniques. These form the class of global methods that accomplish the mapping to visual space by finding a minimum for an energy function, usually called the *stress function*. This function captures the difference between distances in the original space and projection space, or how faithfully the projection preserves distances. In general, however, optimization methods are computationally expensive ($O(m^2)$), even when using efficient numerical solvers [29]. To reduce computational costs, Pekalska *et al.* proposed to select a subset of k observations, called *samples*, and embed these first in the visual space by the optimization of a suitable stress function. In the second step, the remaining data points are placed around the samples, using a global linear mapping. The end-to-end result is an $O(k^3 + km)$ algorithm [171]. Although more efficient than other optimization-based methods, this approach is not fast enough to support interactive applications. Moreover, this method requires a minimum number of sample points equivalent to the dimensionality of the input data.

Another global method composed of two steps is the Least Squares Projection (LSP), which is also based on a non-linear scheme to first position a sample subset in the visual space, and maps the remaining instances through a Laplacian-like operator, yielding an $O(k^2 + m^2)$ algorithm [163]. LSP is based on a global neighborhood graph from which a large sparse linear system is constructed. One important aspect of LSP is that it permits the user to adapt the shape of the projection by changing the positions of the samples in the visual space. However, LSP's global properties put an upper bound to how much and/or how freely one can change a projection in this way. This limitation can be also noticed for in a the PLMP linear mapping method (whose complexity is linear), that also uses a subset of samples to define a global linear projection [164]. Similar to Pekalska's approach, PLMP also requires a minimum number of samples in order to accomplish the projection, which can adversely affect interactivity.

Local methods aim to relax the restrictions imposed by globally projecting all points. Intuitively, they can be thought as 'splitting' the space to project in many small neighborhoods, located around the observations to project, and treating each such neighborhood separately. This way, different local decisions can be taken, resulting in a better projection (in terms of *e.g.* distance preservation) and/or a faster projection algorithm. Local methods make use of two main ingredients to perform the multidimensional projection, namely, the neighborhood information of each data instance or observation, and the location of a subset of samples positioned *a priori* in the visual space. More specifically, the mapping of each data instance depends only on the sample points in its neighborhood, which gives the local nature of the projection process.

In this class of methods, the approach proposed by Chalmers [34] and its hybrid variants [106, 144, 214] first maps the chosen sample subset to the visual space through a force-based scheme inspired in an analogy between stress function minimization and mass-spring systems used to construct graph layouts [56].

The neighborhood structure of each instance is then leveraged to embed the remaining data in the visual space, resulting in an $O(cm)$ technique, where c is the number of iterations performed by the algorithm and m the number of observations in the input data. Several computational optimizations have been proposed in this context, such as using parallel implementations using general-purpose graphics hardware (GPU computing) [73, 98]. However, despite these optimizations, this family of methods is still too slow for interactive applications that deal with large data sets.

The Piecewise Laplacial Projection (PLP) of Paulovich *et al.*[167] uses a force-based scheme to place the subset of samples in the visual space. The remaining data instances are projected using several local Laplacian-like operators, which are built from disjoint local neighborhood graphs. The ease and flexibility offered to the user to interact with the projection is the main quality of PLP – indeed, moving sample points around directly changes the projection layout, which can help to organize (group) similar data points. Drastic changes in the projection are possible with PLP because the underlying local neighborhood graphs are rebuilt during user interaction. However, the continuous changing of the local neighborhood graphs increases PLP's computational cost and can produce rank-deficient local Laplacian systems, which impacts the method's robustness.

The recently published LoCH [68] method obtains good results in preserving the neighborhood distance structures in high-dimensional sparse spaces. It works placing each point close to the convex hull of its nearest neighbors. Comparing to most projection techniques, LoCH is significantly better in segregating groups of similar instances and defining borders between them. This supports, for instance, tasks that involve browsing image collections searching for similar images (Fig. 2.9). The authors of this method also argue that local projection techniques outperform global methods on segregating groups of similar instances and on preserving neighborhood relationships.

Assessing projection quality is a key ingredient in the design of effective visualization applications that use multidimensional projections. Indeed, as compared to earlier visualization methods for multidimensional data, projections (a) create a far more abstract view, and (b) perform a significant amount of data reduction from the original high-dimensional space to the low-dimensional (visual) space. The central (and, in many cases, only) visual element that users can employ to reason about the resulting projection is the distance between projected observations. As such, one has to be sure that this distance correctly reflects the original distance in the high-dimensional data. If this is not the case, then interpretation errors are very likely, *e.g.* incorrectly assessing the similarity of several data points, or incorrectly reasoning about groups, trends, and outliers.

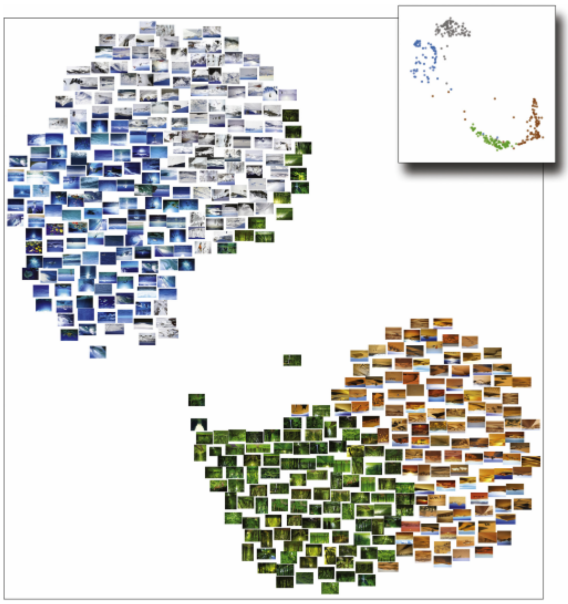


Figure 2.9: The top-right window presents the initial point cloud projection generated by LoCH from a image dataset collected from the internet [68]. The larger image have its thumbnail image instances corresponding with the same point location.

Arguably the best known, and most often used, measure for projection quality is the so-called normalized stress function

$$\sigma = \frac{\sum_{1\leqslant i\leqslant m, 1\leqslant j\leqslant m}(\delta(\mathbf{p}_i, \mathbf{p}_j) - d(\mathbf{q}_i, \mathbf{q}_j))^2}{\sum_{1\leqslant i\leqslant m, 1\leqslant j\leqslant m}(\delta(\mathbf{p}_i, \mathbf{p}_j))^2}, \tag{2.6}$$

where $\mathbf{p}_i$ are the input m observations, $\mathbf{q}_i$ are their corresponding low-dimensional projections, δ is a distance metric for the input high-dimensional space, and d is a distance metric for the output (embedding) space. Typically, for both δ and d, the Euclidean distance is used. Low stress values indicate good distance preservation, the ideal value being zero. High stress values indicate that the projection does not preserve distances well.

Normalized stress is useful to globally quantify projections in terms of their ability to preserve distances. This supports tasks such as *e.g.* ranking different projection algorithms in terms of their overall quality, and judging if a given projection algorithm yields sufficient quality for a given input dataset. However, normalized stress is an aggregated metric which does not give fine-grained insight on projection errors, *e.g.* show which distance ranges are well preserved and which not. Such insight can be obtained by showing scatterplots of the values δ and d for all pairs of points $(\mathbf{p}_i, \mathbf{p}_j)$ in the input dataset. Figure 2.10 shows a few such scatterplots used to compare the PLP, PLMP, and Fastmap projection techniques for three datasets (*wine red*, *segmentation*, and *wdbc*) [167]. If distance is perfectly preserved, such scatterplots should match a diagonal line. Spreads of points above the diagonal indicate that the projection places points too far away from each other (as compared to their original distances). Similarly, spreads of points below the diagonal indicate that the projection places points too close to each other. While stress scatterplots are a useful instrument of getting more insight in the magnitude, spread, and kinds of errors that a projection creates, it still does not directly tell end users where these errors occur in the projection, and thus where misinterpretations can occur. We address these limitations by presenting new ways of exploring projection errors in Chapter 4.

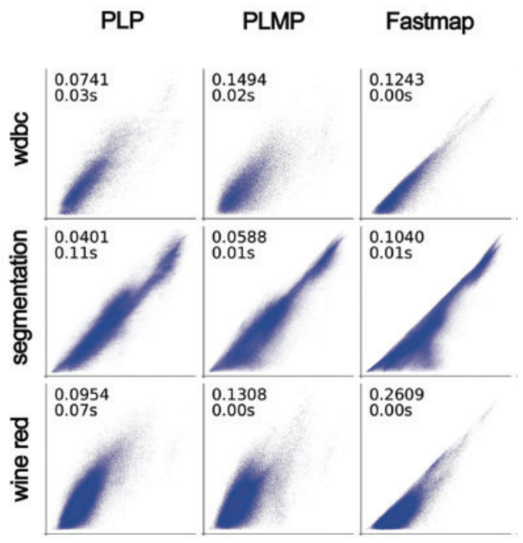


Figure 2.10: Comparing projection quality with stress scatterplots. The x axis maps the distance δ in the input high-dimensional space. The y axis maps the distance d in the embedding low-dimensional space. Image taken from [167].

To better understand quality issues beyond the aggregate insight provided by the stress function, Paulovich *et al.* [168] presented a comparative analysis between LSP, PLMP and LAMP. Their study considered the user manipulation of control points to show different behaviors between global projections (LSP and PLMP) and local projections (LAMP) for a dataset of 2100 observations and 19 dimensions. When 10 percent of data are taken as control points, global and local methods behave similarly and follow the groups' layout created by the user. However, using 19 control points, only LAMP could still separate groups, while the LSP and PLMP could not. Thus, this flexibility to steer the projection layout to closely follow the user layout of the control points is not the only advantage of LAMP. In terms of accuracy, original distances are well preserved in the visual space when comparing with others multidimensional projection techniques. More details about LAMP will be discussed in the next chapter.

Besides stress and its derivatives, there are other quantitative measures that have been used to evaluate the mappings after interactive control points manipulation, namely, neighborhood hit [163], neighborhood preservation [165] and silhouette coefficient [211]. Given an instance x, neighborhood preservation gauges the percentage of the k-nearest neighbors of x that still remain neighbors in the visual space, while neighborhood hit take the k-nearest neighbors of x, checking what proportion of those belong to the same class. The silhouette coefficient, which was originally proposed for evaluating clustering algorithms [211], measures both the cohesion and separation between grouped instances. The cohesion a_x of x is calculated as the average of the distances between x and all other instances belonging to the same group as x. The separation b_x is the minimum distance between x and all other instances belonging to other groups. The silhouette of a projection is given by $Silh = \frac{1}{n} \sum_{x \in \mathcal{X}} \frac{(b_x - a_x)}{max(a_x, b_x)}$ where n is the number of instances. Notice that $Silh$ ranges in the interval $[-1, 1]$ and the larger the value of $Silh$ the better is the cohesion and separation.

Exploring projections: A projection, in itself, can be the final result in several contexts. For instance, in machine learning, projections are used to pre-process very high-dimensional datasets, in order to extract a smaller number of relevant dimensions, also called features. These features are next used to *e.g.* analyze or classify the data at hand. This dimensionality reduction can increase the robustness, accuracy, and running times of subsequent classification algorithms.

In exploratory scenarios, however, a projection by itself is not the final result. The projection is subsequently explored in order to learn insights about the data at hand. For this, additional mechanisms are needed to *explain* the patterns visible in a projection (clusters, outliers, trends). The simplest and most frequently used explanation comes in the form of visually encoding one or a few data attributes per projected point, by means of color, size, and shape. However, this method does not scale to show tens or even hundreds of attribute values atop of each data point. When dimensions have been computed as features, or descriptors, from some higher-level original dataset, such as images, the original data items can be drawn in the form of thumbnails instead of points in the resulting scatterplot [167]. This allows next users to examine why different data points are similar. However, this approach does not scale well with the number of observations. To address scalability, projections can be explained at the coarser level of point clusters rather than individual observations. Clusters are found in the embedding (visual) space, by using any suitable clustering algorithm. Next, the most salient characteristics of each cluster are computed, *e.g.* variables which make the points in the cluster similar, and their ranges, by statistical analysis. Finally, cluster outlines are drawn, *e.g.* by using convex hulls, and the detected characteristics are drawn atop of the clusters as text labels. An example of this approach is given by the recent *ProjCloud* visual data-mining tool [169]. Here, a set of observations, describing news in a collection of text documents, is projected to a 2D point cloud; next, the bisecting k-means clustering technique is used to create separate clusters of similar observations; finally, each such cluster is visualized by rendering its most frequently encountered terms using the well-known tag-cloud technique (Fig. 2.11).

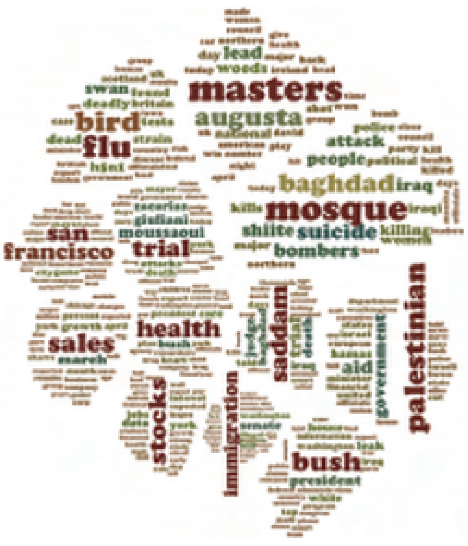


Figure 2.11: Visualization of a collection of news using the *ProjCloud* tool [169].

Biplot axes offer a different kind of explanation: They show the directions and spread, in projection space, of the variations of the original m variables, and they help users finding *e.g.* variable correlations and independent variables. For 2D projections created by PCA from categorical datasets, and visualized with scatterplots, Broeksema *et al.* propose an explanation of the x and y scatterplot axes in terms of the amount of spread of the original m variables along these axes, and visualize these explanations by means of barchart axis legends [28]. While such legends give an intuitive way to understand which variables one can best see along the x and y screen axes, their construction requires a linear projection.

As explained above, projections are typically used to reduce dimensionality, and thereby make it possible to visually explore the data structure in a low-dimensional space. However, one challenge for the resulting scatterplots is that it does not show the original data dimensions. To alleviate this, *inverse* projection techniques can be used. Given a direct projection $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$, an inverse projection is essentially a function $f' : \mathbb{R}^n \rightarrow \mathbb{R}^m$ which aims to minimize the distance $\delta(f'(f(\mathbf{x})), \mathbf{x})$ for all points $\mathbf{x}$ close to the high-dimensional dataset that is projected. One such inverse projection technique is iLAMP, which is based on the LAMP direct projection technique [61]. Inverse projections can be used to extrapolate from the projected scatterplot points, *e.g.* to explore parameter spaces in optimization problems.

A different kind of projection explanation is proposed by da Silva *et al.* [49]. Here, for each neighborhood of the projected points, a ranking of the m variables is computed to reflect their importance in terms of making the respective points close to each other in the original space. Next, the top most important variable per projected point is determined; and the variables which are ranked as top for most of the projected points are determined. Finally, points are colored to reflect the identity of their top-ranked variable, by using a categorical colormap. The technique can be also applied to explain the positioning of each point in terms of several variables rather than a single one. The resulting visualization implicitly splits the scatterplot into several compact same-color areas, thereby explaining the projection in terms of a few variable identities.

2.4.4 *Other Approaches*

Hierarchy-based approaches, tabular displays (Heatmap), glyphs and animation are some examples of visual mappings that also deals with multidimensional data and they are quite well known by the academic literature.

Chernoff's faces approaches the visualization of multidimensional datasets by extending the concept of glyphs, used since long to visualize scientific data such as vector and tensor fields [217]. A glyph, in this context, is a 2D or 3D parameterizable object, whose m parameters are controlled by data attributes. For example, vector fields can be visualized by encoding their direction and magnitude in the orientation, respectively length, of an arrow icon. This idea can be generalized to higher-dimensional datasets, as long as one can design a glyph having m perceptually independent visual attributes. Chernoff proposed, to this end, to use human face-like shapes, based on the assumption that such faces have many independent dimensions (when perceived by users) [42]. However, such visual attributes as offered by parameterized faces are not necessarily of ratio type, to follow the attribute taxonomy outlined in Sec. 2.2.1. Moreover, the number m of independent visual attributes is much lower than the hundreds, or even more, attributes present in many multidimensional datasets. Similar ideas are proposed by the stick figures in [174], where an articulated line figure is used to map two attributes to the figure's position in 2D and additional ones to the rotation angle, length, color, and thickness of the figure's limbs. Compared to Chernoff faces, more figures can be packed on the same screen space, thus more observations can be visualized simultaneously. However, the mapping of attributes to the figure's degrees of freedom has to be done with great care to avoid generating cluttered, thus meaningless, images. Shape coding [14] and color icon [129] techniques also propose a glyph-based approach, where each observation is encoded into a small rectangle where the color of each pixel represents the value of a separate attribute. However, detecting patterns that span several attributes can be hard, so the scalability of these techniques in terms of number of attributes they can depict is limited. For more details on these and other multidimensional data techniques, we refer to the recent survey in [135].

2.5 EXPLORING MULTIDIMENSIONAL MULTIMEDIA DATA

One application domain that generates high-dimensional data that is not necessarily spatial, and contains attributes of various types (following the taxonomy in Sec. 2.2.1), is the *multimedia* domain. Loosely put, this domain encompasses the generation and exploration of large collections of data (also called content) of multiple *media* types, such as text, graphics, still images, full-motion video, sound (music and voice), and animations [23]. Technological and societal developments over the past decade, such as the proliferation of low-cost audio-video acquisition devices such as smartphones and webcams, the increasing availability of broadband internet and low-cost data storage in the cloud, and the appearance of social networks such as Facebook or Instagram, have made a huge amount of multimedia content available. As a consequence, both researchers and commercial parties are keen to develop efficient and effective techniques to explore these large information spaces.

Following our multidimensional data characterization introduced in Sec. 2.2, an observation in this context can be seen as one set of related content elements which has been authored, or created, as a single entity, *e.g.* a movie containing one or several audio tracks, one or several subtitle tracks, and a video track. Such an observation can be further described in terms of a so-called set of *features*, also called a feature vector [191]. These are typically measurements performed on the content data, with the aim of quantifying various properties of interest. When such features are extracted jointly from different media types, such as text, audio, and video, the resulting feature-set is typically called multimodal [85]. Examples of features are topics extracted from text; keyframes denoting important action points in a video stream or optical flow descriptors extracted from motion video; color moments, texture descriptors, and shape descriptors extracted from still images or individual video frames; and sound descriptors such as loudness, pitch, and spectral descriptors extracted from audio data. All such features have the advantage that they can be described by real numbers. As such, an entire multimedia dataset can be modeled as an observation point in $\mathbb{R}^n$.

Multimedia datasets are a very interesting case to test the effectiveness of multidimensional data analysis and data visualization methods for several reasons:

- They easily yield several tens or even *hundreds* of dimensions (if we consider the large set of features one can extract from audio, video, and subtitles);
- The extracted features are highly *abstract* (*e.g.* consider edge histograms extracted from image data);

- The extracted feature *types* span the entire spectrum of attribute types known in information visualization (continuous, discrete, ordinal, categorical, relational, and text);
- The number of *observations* is very high – one observation per frame can, for a typical movie recorded at 24 frames-per-second, create tens of thousands of observations;
- Multimedia data admits a multilevel *hierarchical* structure – consider *e.g.* a frame, part of a soccer match video, part of a tournament match-set;
- Multimedia data are literally everywhere, created in huge amounts, and of great interest to a wide range of users, from professionals like trainers analyzing athlete behavior to casual consumers like fans browsing a soccer match collection.

Considering that multimedia data offers an interesting, challenging, and valuable testing-ground for multidimensional-data visual exploration techniques, we next overview the most prevalent analysis and exploration tasks, and related supporting visualization tools, that target multimedia content, with a focus on video data.

2.5.1 *Video analysis steps*

Video is a communication tool that gained popularity with the appearance of cinema theater and television devices in the twentieth century. With the digital revolution, technological advances and the increase of internet access, among other factors, have made video a key component of information dissemination. In turn, this has led to the development of applications in many domains, such as education (online courses), surveillance and security, entertainment, and news. Two important challenges in this context are the *extraction* and *presentation* of information from video content, so that analysts can quickly and easily form an impression about the content represented in the video.

According to Hanjalic [85], the scope of video content analysis is the extraction of information about the content conveyed by video data. In this sense, video-analysis algorithms and techniques aim to enable users to quickly access events, persons, and objects captured by the camera, and also to efficiently generate overviews, summaries and abstracts of large video collections. Video content analysis algorithms can be further refined into three subclasses:

1. *video parsing* refers to the temporal segmentation of the video into fragments that describe events at desired levels of detail. Examples hereof are individual video shots (low-level level of detail) and scenes (high-level level of detail);

2. *video indexing* refers to the assignment of links, or indexes, between a specific semantic category, *e.g.* a categorical description of the type of action represented by a video segment, and the respective video segment. Once a video is indexed, it can be analyzed on a high level, *e.g.* queried for the inclusion of specific events, based on desired semantic category search terms;

3. *video abstraction and representation* refers to the generation of compact and comprehensive representations of video data based on indexes assigned to parsed video fragments. This step aims to communicate the 'essence' of a video dataset to users in a more efficient and effective way than what users would typically obtain by plainly watching the entire video.

The three above-mentioned groups of technique essentially create a so-called multiscale representation of the video data. On the lowest level, shots and scenes can be seen as syntactic elements; the intermediate indexing level can be seen as associating (basic) semantic attributes to syntactic elements; the last level conveys the highest-level representation, where a 'story' can be assembled from the semantic elements. The challenge in this respect is finding ways to cover the gap between the low-level feature vectors extracted from parsed video data and the high-level events that one wants to identify and reason about. This process is hard since there is no unique, context-independent, mapping of low-level (syntactic) data to higher-level (semantic) events. This mapping can be video-dependent, task-dependent, and person-dependent. Performing this mapping is also known under the name of bridging the 'semantic gap' [206]. In order to reduce this gap and facilitate the identification of events, researches started to analyze videos with techniques that are specific to the domain the video is related to. In this sense, Snoek and Worring [208] propose ten different domains for video and their related semantic-index hierarchies: talk show, music, sports, feature film, cartoon, soap, documentary, news and commercial.

We will next focus the discussion in the video content analysis related works on techniques that are applied to either identification but mostly visualization of video segments in different types of domains.

2.5.2 *Analysis and presentation of video content*

The tasks of analyzing video data to extract relevant information and presenting this information in effective ways to the user are intimately connected. This is, on the one hand, due to the fact that specific analysis methods are needed to extract information that is needed to support specific user tasks; and, on the other hand, that such specific information requires customized ways to present it to achieve maximum effectiveness. As such, we will next jointly discuss methods for video analysis and presentation.

The main four types of techniques addressing analysis and presentation of video content are video summarization [5], video abstraction [221], video browsing [196] and video visualization [22]. They can be described as follows:

1. *Video summarization* uses content-based analysis techniques to create video summaries, which aim to intelligently subsample a video stream so as to preserve events or timespans of interest, and compress or even remove the rest. From a data processing perspective, summarization can be thought of as a non-uniform, semantic subsampling process that tries to maximize the information amount captured by the samples used to represent the video. Summarization is used in *e.g.* video retrieval systems. A recent review identified 11 different application domains of summarization techniques (including news, sports, traffic, surveillance, documentary, and music), and showed that not all techniques are equally good summarization solutions for all domains [5].

2. *Video browsing* aims at offering efficient ways for users to explore (large) individual videos or video collections. Video browsing can be further classified in three approaches [196]: Videoplayer-style browsing aims to improve classical videoplayer-like user interfaces; browsing by video retrieval querying aims to offer compact and comprehensive ways for users to understand the types of videos present in a large video collection, by *e.g.* using use storyboard techniques; finally, browsing through video surrogates conveys video content information at higher, and typically more abstract, levels than images. This increases the amount of video information that can be browsed within a limited amount of time and/or by using a limited amount of screen space.

3. *Video abstraction* is used to create short and compact summaries of videos, which can next be used for tasks such as summarization or browsing. Summaries can be either sequences of static keyframes or short sequences of animated images, also called movie *skims* [221]. Keyframe-based abstraction is relatively simple to implement and computationally efficient, but asks more effort from the user to grasp the summarized content. Video skims can capture more content and are also found more visually pleasing, but are harder to compute while maintaining context and coherence.

4. *Video-based graphics and visualization* focuses on presenting information extracted from videos in novel ways [22]. Video graphics create and render graphical models built from video data, with the aim of creating novel digital content for consumers. In contrast, video visualization creates new visual representations of the video data (or of artifacts extracted from such data), aiming to help users find and analyze important features or events, reduce the time needed to watch videos, and ultimately gain insight to make decisions in a cost-effective manner. Both techniques can be thought of as being generalized forms of video abstraction.

As mentioned earlier, video analysis and presentation solutions are typically developed in strong connection with the targeted user group. As user groups can be, next, associated to different types of video content, it is appropriate to next discuss such existing solutions by classifying them based on the content type. In this respect, we find three important (though not exhaustive) categories of content type and related video visualizations: Sports videos, movies, and TV news. These are discussed next.

Sports videos encompass a wide range of content related to sports events, *e.g.* recordings and live broadcasts of individual matches, and collections of temporally or hierarchically organized videos describing entire tournaments. General-public tools for visually exploring sports data also include the broad category of infographics used in printed and/or online forms by many newspapers. However, we will focus here on the more advanced, albeit still aimed at the general public, tools for sports data visual exploration.

In this domain, the quest for appealing and easy-to-understand visual presentations of videos is well recognized [157, 9, 179, 175]. The importance of team sports is noted in a visualization model built based on a literature review, classifying sports visualizations among conceptual axes [157]. For American football, a visualization that showed the trajectory of the ball during the game was proposed [9], using refined ball-tracking techniques to avoid problems with camera and radio-based tracking due to occlusion or ball proximity to groups of players. Eight receiving antennas captured multidimensional motion data, which was filtered, amplified, sampled with a bank of analog-to-digital converters, stored for post-processing and merged with the video to determine the position and trajectory of the ball. While this visualization provides ball-position estimates to the spectator, tracking is done per-video-frame, so contextual event information is not present. In contrast to such per-frame tracking, TenniVis takes an opposite approach [179]: Statistical information, extracted from the video stream, is mapped atop of each video segment, or video skim, using glyphs, pie charts, and bar charts (Figure 2.12). This information comes from a multidimensional dataset composed of the following attributes for each game point: start and end time; who won the point; the type of outcome (ace, double-fault, winner, forced error or unforced error); and service faults (missed serves). While this approach is good in showing aggregated performance metrics, lower-level, event-related, insight is lost; separately, some of the used graphics may be found too abstract by casual users. For ice hockey, the SnapShot system maps multidimensional data in a 15 shot attributes like length, goal or not, shooting team, shooter's name and location in a single view [175]. A novel radial heat map was developed to visualize the distance of shots. As for TenniVis, we argue that such visual presentations are useful for advanced users such as analysts, but too abstract and/or complicated for the simpler questions of casual sports fans.

Summarizing the above, sports videos are a challenging analysis domain as they provide high-variate, high-volume, and hybrid data consisting of the posi-

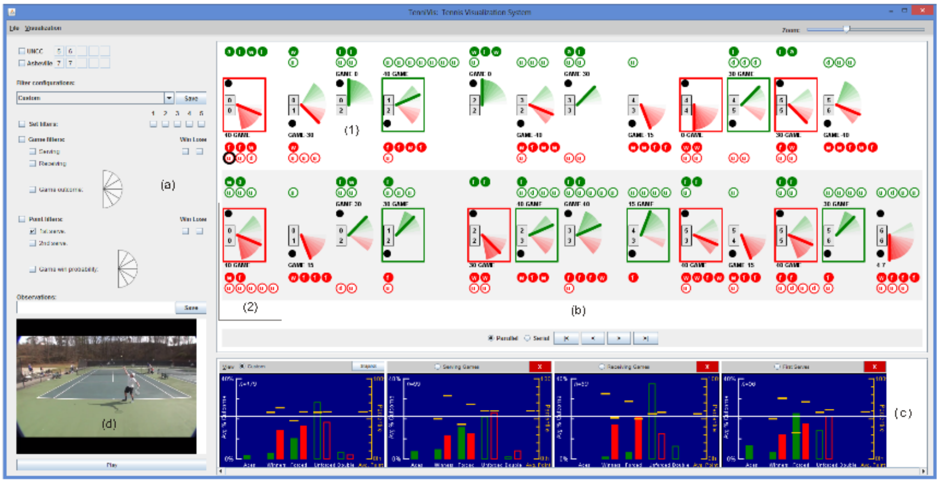


Figure 2.12: Visualization of a single tennis match in TenniVis [179].

tion and trajectory of the ball and several players; labeled moments describing important game events; per-game aggregated statistics such as ball posession, number of faults, and final score; and tournament-level information such as *e.g.* the hierarchical organization of matches. Users targeted by sports video visualizations come from two main categories: Trainers and other sports specialists need to analyze high-fidelity, fine-grained, information that captures game patterns, tactics, and strategies, and are thereby more open to the adoption of complex visualization techniques. In contrast, casual consumers, such as sports fans, require a very low learning-curve, intuitive interfaces which are usable on a wide range of consumer electronics devices (*e.g.* set-top boxes, TVs, and smartphones), and are more interested in spotting and viewing salient match events, such as goals or controversial sequences rather than in in-depth tactical analysis.

Movies and TV series are another prominent type of video content. For this domain, most of current work related to video visualization goes in the of summarizing movies by showing their narrative or story in a compact visual way [45, 38, 246, 134]. Correa and Ma [45] proposed an appealing linear collection of mosaics, defined as panoramic summaries of short video sequences occurring over a common background [45]. The mosaic collection is also known under the name of *dynamic narrative*. This type of summarization offers users the possibility to dynamically explore a movie video, or video collection, by browsing/watching a sequence of short video clips. The method use SIFT local feature descriptors [136] to create multidimensional data for next finding matches between adjacent movie frames, and a Gaussian Mixture Model to identify if a pixel is part of the image background or foreground. For movies such as the “Superman” cartoon and “Happy Go Lovely”, this technique was shown to

achieve good results. However, it was also noted that this technique has limited abilities to determine the foreground in fast camera motions and also detect objects atop of a moving background. Visual Storylines [38] builds a geometric layout to present sub-stories in a movie. Shot dissimilarity is calculated in a feature space using image features (color histograms), audio features (Mel-frequency cepstral coefficients (MFCC) feature distance [52]), and temporal constraints. For some movies, such as "Star Wars: Attack of the Clones", the method provides good results (Figure 2.13). However, complex interaction within character groups showed to be one drawback. Audio and v features are also used to construct a so-called Affective Visualization for movie browsing [246]. This approach used 22 audio features and 5 visual features to describe the movie and extract its video segments. Drama, horror, comedy, and action were the movie categories tested in the visualization tool, which aimed to detect and highlight higly-emotional moments in the movie. In this respect, the proposed technique showed some limitations for drama-type content, since in this type of content emotions are more likely expressed by conversations between characters (rather than salient audio or video patterns), and the semantics of such conversation is hard to capture by low-level audio-visual features. The Affective Space component maps the detected video segments to points in a point cloud or scatterplot-like visualization, according to the similarity of the so-called 'affective state', a four-valued attribute describing the emotion captured by a video segment. This enables users browsing video segments which are similar from the perspective of the detected affective state, but looses the temporal coherence between such segments. Instead of presenting frames from the movie, StoryFlow [134] illustrate the dynamic relationships between entities in the movie story by drawing adjacent lines. Line bundling techniques can be next used to minimize the visual clutter caused by many lines crossing at various places and under various angles, and thereby create a high-level display of the dynamics of relations. Similar edge bundling approaches have been used, in a different context, to create scalable displays of dynamic graphs in information visualization and software visualization [95, 94].
TV news represents a separate sub-domain, related to videos that contain news broadcasts. Here, visualization techniques have the main aim to preserve and emphasize the specific information related to the news event being highlighted in the broadcast. We note that, in contrast to typical movies and sports events, news are much more densely packed with information – consider for example a typical five-minute video fragment from CNN or CNBC, showing several presenters in multiple views, several stock-exchange scrolling banners, and optionally additional timelines, bar charts, weather alerts, or similar infographics.

Several summarization methods have been proposed that use static images, or frames, to represent the respective shots in a single video [76] or in a video collection [138, 139, 241]. MediaMill [241] is a semantic video search engine that shows different shots aligned in the display space. Closer to our interest, other approaches convert the multimedia data into a high-dimensional dataset,

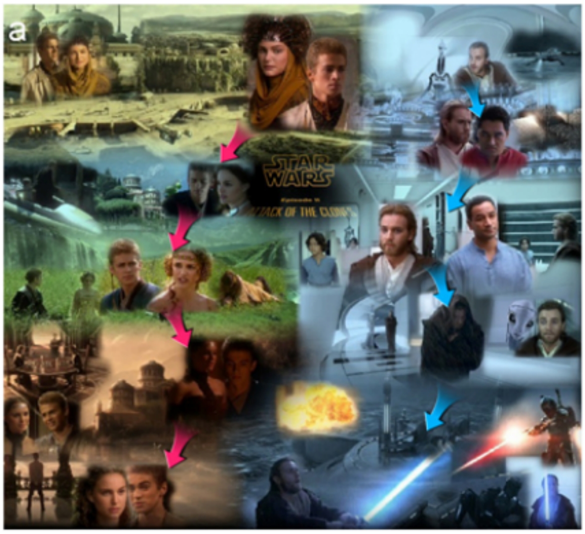
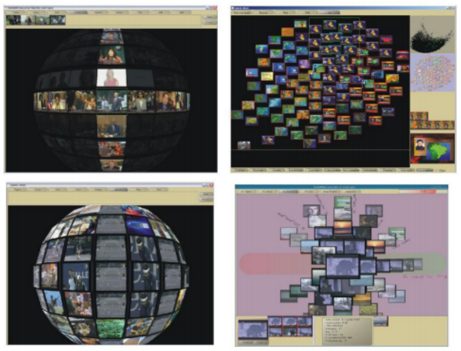


Figure 2.13: Visual Storylines [38]

by extracting features such as text from audio (by using speech recognition), or low-level audio features such as cepstral coefficients and derivatives, zero crossing rates, and bandwidth; and features from video such as Wiccest and Gabor filter responses for each video shot. In addition to these audio-and-video features, time and semantic information can be added to complete the high-dimensional dataset representing the input multidmedia data. Such datasets can be next explored by several visualizations. Figure 2.14a shows four such visualization examples: CrossBrowser correlates text extrated from speech (vertical axis) with time (horizontal axis); SphereBrowser maps semantic concepts on the vertical axis and time on the horizontal one. RotorBrowser maps visual similarity and semantic concepts in the same visual space; and GalaxyBrowser uses a 2D projection of the multidimensional dataset, constructed by using the ISOMAP projection technique [218] to map visual similarities of the recorded data points (video shots).

Luo *et al.* used the spatial position of the selected representative video shots to convey additional semantics in the video visualization [138]. For this, they first extract a so-called interestingness metric from news videos, which aims to capture the relevance of the presented news for the average watcher. Interestingness is measured by combining features extracted at different levels, such as visual descriptors of shots (local color histograms); and semantic concepts, representing keyframes and keywords, respectively. The proposed visualization uses images organized in five columns containing keyframes of three possible sizes. Keyframe size is used to encode the interestingness value. The five-column layout follows a semantic zooming approach, where the middle column repre-


(a) Browsers in the MediaMill search engine [241].

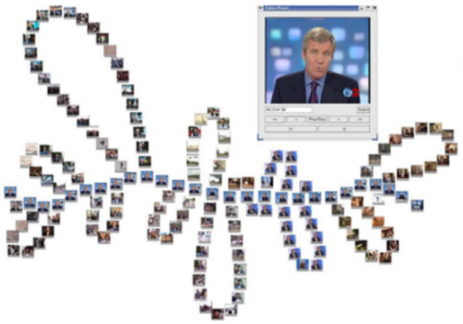

(b) Table of Video Content visualization [76].

Figure 2.14: Visualization examples for TV news content.

sents the focus (displaying only 5 keyframes); the left and right columns from the middle represent one detail level further (displaying each 7 keyframes); and the outermost two columns represent the coarsest level of detail (displaying each 11 frames). Luo *et al.* next extended their visual exploration and interestingness metric to address visual exploration of news and relations thereof [139]. For this, a graph representing relations between news events is first created, using keywords and keyframes as nodes, and their relations as edges, respectively. Edges are weighted according to their measured interestingness. This way, highly interesting *and* related news events will naturally cluster together in a layout of the above graph. The resulting graph can be next explored using traditional hyperbolic views which highlight a user-chosen region of interest (*i.e.*, related-news group). The proposed visualization was tested by exploring news networks mined from three different channels (CNN, FOX and MSNBC), and was found effective and efficient in conveying a 'map of the news' type of insight to both casual users and users having specific interests in terms of news types to examine.

Finally, we mention the interesting approach of Goeau *et al.* for the browsing of a wide spectrum of TV programs, including news, sports, and magazines [76] (see also Fig. 2.14b). Their technique, called Table Of Video Contents (TOVC), gives a global overview of a video showing temporal and structural information. For this, a visual similarity between shots is calculated based on features such as color histograms in the RGB, HSV, and LUV spaces; histograms of luminance edge orientations (local gradients); and affine motion estimated from tracking salient image features found by corner detection. Next, projection techniques using multidimensional scaling (MDS, see Sec. 2.4.3) are used to reduce the dimensionality of the obtained feature space. The obtained information is visualized along a 1D structure, called backbone, which represents the chronological order of extracted video events. This backbone is annotated with information representing the anchors (TV presenters), loops (representing report sequences or interview shots), annotated by their duration; and color-coded semantic tags extracted from the analyzed video, *e.g.* blue to indicate indoor studio shots, and red and yellow colors to indicate disorder events such as riots or fires.

2.6 CONCLUSION

Upon a global analysis of the various aspects of the related work outlined above, three aspects become salient, as follows.

Multidimensional exploration challenges: Multidimensional data are becoming increasingly more prevalent, and more important, in many application domains. However, in the same time, current state-of-the-art visual exploration techniques for such data are limited in many respects – number of observations they can handle; number of dimensions they can handle; types of data attributes that are supported; types of questions these visualizations can answer or, more generally, types of data-related patterns they can highlight.

Multimedia sports data: The domain of multimedia data represents an excellent test-field for visual exploration methods for multidimensional data. Multimedia data are huge, easily available, can be described by tens up to hundreds of attributes of different kinds (image features, audio features, text topics, semantic labels), comes with a natural hierarchy, has the additional challenge or being time-dependent, and – last but not least – is considered interesting for a large and diverse palette of users ranging from professionals to casual consumers.

Within the multimedia domain, *sports* datasets form a particularly interesting sub-domain. On the one hand, the share the general characteristics of multimedia data outlined above (size, dimensionality, hybrid attributes, time-dependent data, wide user range). On the other hand, sports data are nicely located roughly at an average 'level-of-information-density': Compared to typical movies, sports videos have much more clearly defined quantitative information and discernable structure, *e.g.* they have a clear notion of a fixed number of players, play rules, score points, and structured spatial motion over a clearly-defined range. Compared to TV news, they have a more fuzzy and also more dynamic character, in terms of motion dynamics and nonlinearity of the underlying story. Separately, we observe that the current state-of-the-art of multimedia visual-exploration techniques has only scratched the surface of the possible in terms of extracting and depicting patterns of interest and also in terms of developing effective metaphors for presenting this information conveniently to different types of users. As such, we believe that sports videos, and in particular those describing team sports having multiple interacting players, are a good candidate for testing new multidimensional visualization techniques.

Potential of projections: In the context of multidimensional data visualization techniques, *projections* are an interesting and important contender for a scalable, generic, and general-purpose technique. Indeed, projections can easily handle tens up to hundreds of thousands of observations and hundreds up to thousands of dimensions, in terms of computational and visual scalability; they can handle any attribute types, as long as a suitable distance metric in attribute space is provided; and they can work, in general, automatically, with little user intervention in terms of parameter tuning. However, on the downside, projections generate too abstract depictions of data, even for trained users. For instance, they are good in showing groups of similar observations, but not in showing *why* these observations are similar, or how to read the values of *attributes* from the projection.

Given the above, the focus of this thesis will next be on two main questions:

- How can we increase the *usability* of projection techniques to offer intuitive, easy-to-use, customizable views on multidimensional datasets to a large spectrum of users;

- How can we use projections and/or additional visualization techniques to allow the same large spectrum of users to explore *multimedia* collections, such as sports games, in effective and efficient ways.

Both above issues will be detailed in the upcoming chapters.

3 LOCAL AFFINE MULTIDIMENSIONAL PROJECTION

ABSTRACT: *Multidimensional projection techniques have been an interest research topic in visual data mining, mainly regarding the evaluation and effectiveness of such techniques. In the context of their application scenarios, there is a lack of fully interactive mechanisms to handle complex data, such as multimedia content. This work presents a novel multidimensional projection technique called Local Affine Multidimensional Projection (LAMP) developed to be more flexible and versatile than existing projections in the same class, and which provides to the user the possibility to dynamically modify local transformations according to his/her knowledge. Its computational efficiency is demonstrated by a set of comparisons against other multidimensional projection techniques. Its versatility is demonstrated through applications involving text documents and multimedia data* [1].

3.1 INTRODUCTION

As described in Chapter 2, the increase of big data collections consisting of high number of observations having many dimensions has spawned the development of novel visual exploration methods to make sense of such data spaces. One specific class of such methods are known under the name of multidimensional projections (MPs). MP methods have been used in several visualization-intensive applications involving vector field analysis [51], visual text mining [39, 166], and word cloud formation [48], to cite only a few.

Due to technical advances in scalability and accuracy, MP methods are becoming increasingly more attractive as a building block for multidimensional visualization solutions [164]. However, these methods still bear weaknesses that impair their use as fully interactive visual exploration tools, depending on (1) the data nature or (2) intrinsic method properties. First, computational costs, accuracy and robustness are important when dealing with massive and distinct data, as multimedia for instance, especially when we may need to (re)generate projections in real-time to support interactive exploration. Secondly, drawbacks also relate to the global *vs* local behavior of MPs. Global methods project an entire dataset in a go, which makes them simple to use, but inflexible in terms of locally adjusting the projection in a specific region (subset of observations). Local methods allow specific refinements of the projection in a given area, but can either present high computational costs or do not provide mechanisms flexible and robust enough to permit the user to freely intervene in the projection. This lack of flexibility typically relates to the user having to position and/or manipulate a large number of observations in the visual space, in order to attain

1 This chapter is based on the paper *Local Affine Multidimensional Projection* (P. Joia, F. V. Paulovich, D. Coimbra, J. Cuminato, L. Nonato), IEEE TVCG, vol. 17, no. 12, 2011, pp. 2563-2571. P. Joia mainly contributed to LAMP's technique and theory. D. Coimbra proposed, constructed and evaluated the applications of LAMP.

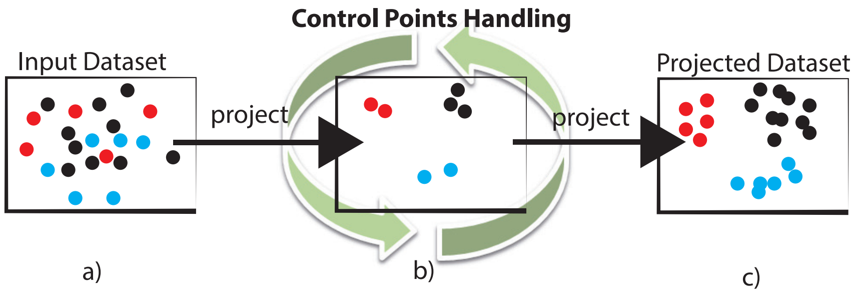


Figure 3.1: User interaction process with MP using control points. a) Considering a input dataset, the control points are projected in the visual space. b) The user freely handling the points in a iterative process. c) Remaining instances are interpolated, taking into account the geometry given by this initial placement, projecting the whole dataset.

the desired projection. All in all, this makes the design of a custom projection a tedious and time-consuming process.

Besides allowing localized control of the projection shape (as their name says), local projections are useful for a higher-level reason – they allow users to insert their domain-expertise in the data exploration and knowledge discovery process. This occurs when the user manipulates a subset of observations projected in visual space, called control points, in order to separate dissimilar observations and group similar ones (see Figure 3.1 for an illustration of this process). The remaining observations are next automatically placed by interpolation by the projection technique, taking into account their similarity with respect to the control points. Given the local nature of the projection, the placement and/or movement of control points only affects the shape of the projection locally, thereby allowing the user to effectively 'design' a desired projection shape by local manipulations, much like designing a 3D shape in computer-aided design by manipulating its control points. At a high level, this process allows users to specify the *desired* similarities between given observation-pairs directly in the projection, and have the system 'learn' this information to compute a suitable projection for all observations. For instance, if observations are images, users can construct an organized map of a large collection of images by simply placing a few control images at desired positions in the visual space, and having the local MP place all other images around these control images according to their similarity.

In this chapter, we present the design of an efficient and effective method, called *Local Affine Multidimensional Projection (LAMP)*, that exhibits the interactive local control of the projection outlined above. The orthogonal mapping theory provides the basis of LAMP's mathematical formulation, ensuring computational effectiveness, such as robustness and accuracy. Moreover, due to its mathematical properties, LAMP offers the possibility to have a reduced number of samples to build the projection mapping. On the usability side, LAMP assures flexibility by allowing a few interactions to incorporate user knowledge into the projection process, allowing a dynamic exploration and organization of data. We have tested such properties in a data correlation application with image and audio (creating multimedia slide shows), followed by a visual exploration of textual documents. These aspects are detailed in Section 3.5.

In summary, the main contributions of this chapter are:

- *Technique*: LAMP is a multidimensional projection technique that derives from orthogonal mapping theory to build accurate local transformations (Section 3.3). LAMP can be tuned to be both a local and a global method, projecting data with high accuracy into the visual space (Section 3.4). Thus, LAMP can be suitable for interactive applications due to the reduced number of control points.
- *Data Correlation Application*: A new visualization-based on data correlation shows LAMP's flexibility and effectiveness. In Section 3.5 this correlation is demonstrated by an application that relates data from distinct data sets by only manipulating control points, creating multimedia content such as slide shows with sounds.

To the best of our knowledge about multidimensional projections context, neither has the orthogonal mapping theory ever been used to this end, nor have such projection techniques ever been employed to interactively correlate instances from unrelated datasets.

3.2 RELATED WORK

As outlined in Sec. 2.4.3, multidimensional projection methods aim at mapping instances, also called observations, from a high dimensional space to a low-dimensional space (typically, the 2D screen space) so as to preserve distances between pairs of observations as much as possible. At a high level, projection methods are challenged by two main issues: (a) computational complexity (typically $O(n^3)$ for n instances), and (b) the difficulty of locally controlling the distribution of points in the resulting projection. To cover those issues, we present next our proposal.

Our proposal: In the context of multidimensional projections state of the art and their inherent issues (Sec.2.4.3), our proposed LAMP method offers a unique combination of desirable properties: In contrast to global approaches, LAMP can be set to behave as a local projection method, thus avoiding most of the problems inherent to global techniques. Its computational complexity is $O(kn)$, as discussed further on in Sec. 3.3. Besides being cost-effective and highly precise, LAMP does not rely on neighborhood graphs. Furthermore, its mathematical formulation admits it to use a very small set of sample instances as input, which makes its use in interactive contexts very effective, as shown later on by the applications presented in Sec. 3.5.

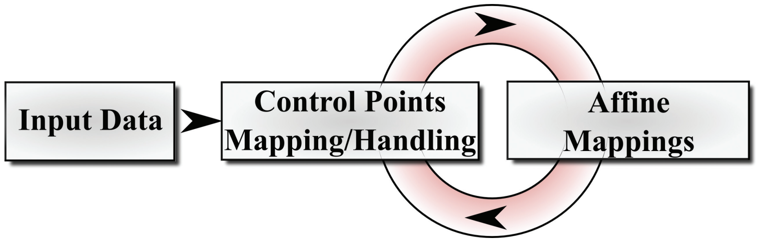


Figure 3.2: Dataflow-like pipeline presenting the components of LAMP.

3.3 TECHNIQUE DESCRIPTION

LAMP builds upon the insight provided by earlier representative-based projection methods, to propose a two-step projection process based on placement of such representatives, followed by placement of the remaining points around the representatives. To follow the terminology in [103], we will further call these representative points *control points*. This is further motivated by the fact that, unlike other methods, we allow users of LAMP to (interactively) move these control points in the projection space in order to control the projection's shape.

Key to LAMP's control of the projection is using information from the control points to build a set of orthogonal affine mappings which follow the control points' layout. As such, we can next use these mappings to interactively steer the projection by simply moving the control points, as outlined in Figure 3.2. If, as we will see, the number of control points can be set to a small fraction of the total point count, and the motion of a control point has only local effect (with respect to instances that are close to that point), this offers an easy-to-use and predictable way to steer the projection's shape.

Section describes next the technical details of LAMP 3.3.1. Next, Section 3.3.2 discusses how control points can be selected for optimal results in practice, and also presents an analysis of stability and robustness of the LAMP method when the number of control points changes.

3.3.1 *Affine Mapping Computation from Control Points*

Let our input dataset that we wish to project denoted by $\mathcal{X}$, with elements $x \in \mathbb{R}^m$. From this input dataset, we next select a set of k control points, which we next call $\mathcal{X}_S \subset \mathcal{X}, \mathcal{X}_S = \{x_1, x_2, \ldots, x_k\}$. In the following, we assume that k, the number of control points, is much smaller than the total number of data points $|\mathcal{X}|$. Denote the i^{th} element, or control point, of $\mathcal{X}_S$, by x_i. Finally, let $\mathcal{Y}_S = \{y_1, y_2, \ldots, y_k\}$ be the projection (placement) of the control point set $\mathcal{X}_S$. Since we are interested in 2D projections, $\mathcal{X}_S \subset \mathbb{R}^2$.

To project a data point $x \in \mathcal{X}$, LAMP aims to find an affine transformation $f_x(p) = pM + t$ that minimizes

$$\sum_i \alpha_i \|f_x(x_i) - y_i\|^2, \quad \text{subject to } M^T M = I \tag{3.1}$$

where the matrix M and vector t are the unknowns to find via optimization, I is the identity matrix, and α_i are scalar blending weights defined as:

$$\alpha_i = \frac{1}{\|x_i - x\|^2} \tag{3.2}$$

Intuitively, we can explain Equations 3.1 and 3.2 as follows: The function$f_x(x_i)$ tells the effect of a control point on an instance x; the squared distance $\|f_x(x_i) - y_i\|^2$ captures how well the projection of a regular (non-control) point 'matches' the projections y_i of control points; and the weights α_i diminish the bias of control points x_i which are far away from instances x, so that the control points have a localized effect (with respect to data instances which are close, in the high-dimensional space, to their own locations in the same high-dimensional space).

It is important to node that the minimization problem (3.1) is related to other optimization applications, such as the "as-rigid-as-possible" image deformation [195]. A main difference, however, is that, while [195] aimed to create a mapping from $\mathbb{R}^2$ to $\mathbb{R}^2$, we now need to address the more complex problem of creating a mapping from $\mathbb{R}^m$ to $\mathbb{R}^2$, where typically $m \gg 2$. This has two effects: First, from a technical viewpoint, we cannot reuse the same explicit minimization formulas proposed by [195], but need to develop a more general minimization process. Secondly, from a practical viewpoint, our minimization is more complex, since there are (obviously) more ways to map points from $\mathbb{R}^m$ to $\mathbb{R}^2$ than from $\mathbb{R}^2$ to $\mathbb{R}^2$, even when constrained by the same usage of affine mappings.

To get more insight in the above challenges, we next outline several important technical aspects involved in our minimization.

First, note that the constraint $M^T M = I$ implies that the resulting affine mapping f_x behaves like a rigid (isometric) transform – that is, it will only rotate and/or translate points, but not introduce scaling and shearing. This reflects our desire to preserve Euclidean distances during the projection.

Secondly, note that if no such constraint is used when minimizing Eqn. 3.1, we can solve this problem by standard least-square fitting techniques. Yet, if we did so, potential errors that occur due to an improper placement of the control points x_i will trickle down via the affine mappings to the placement of the regular points x, leading to lower-quality projections. In contrast, if we use the orthogonality constraint $M^T M = I$, such control-point placement errors propagate *less*. It is important to note that projection errors (with respect to the preservation of Euclidean distance ratios between $\mathbb{R}^m$ and $\mathbb{R}^2$ are, in general, unavoidable, in the case of a general high-dimensional dataset and a free placement of control points in $\mathbb{R}^2$. The orthogonality constraint aims to keep these errors as small as possible, but cannot eliminate them completely. For a deeper insight into the use of orthogonal constraints in minimization problems alike to Eqn. 3.1, we refer further to the book by Gower and Dijksterhuis [78],

Thirdly, we should note that the weights α_i of the control points x_i depend also on the instance to project x. In other words, a separate affine mapping is generated for each instance x as subjected by the k control points x_i (rather than generating k mappings for the control points x_i which would be used to place the instances x). Intuitively, we can say that the regular instances x 'pull' their position information from the control points x_i, rather than having the control points x_i 'push' their influences to the regular instances x.

Finally, in contrast to [195], we do not want (or need) to ensure spatial continuity constraints for the overall mapping (projection). Rather the contrary: We want to allow discontinuities in the projection to take place. These allow more freedom in placing highly-dissimilar data points in $\mathbb{R}^2$ so that we have more leeway to place highly-similar data points close to each other. To achieve this, we will restrict the summation in Eqn. 3.1 only to control points x_i located in close to the current point to project x. Note that this effect is also present in the definition of α_i. Intuitively, this makes the projection behave in a *local* fashion –thereby the L in LAMP. Note also that locality is affected not only by the size of the neighborhoods around control points, but also by the total number k of control points. Indeed: The larger the number of control points x_i (with non-vanishing weights α_i) we use in the summation in Eqn. 3.1, the less local will the projection of x be. Hence, if we increase the number of control points (keeping all other parameters fixed), a given instance x will be influenced by more control points, since the likelihood more such control points fall near it increases. We will refine these observations next in Sections 3.3.2 and 3.4.

Let us now focus on the minimization procedure for Eqn. 3.1. For a minimum point, we have the partial derivatives of f_x with respect to t equal to zero. This allows us to rewrite t as a function of M as

$$t = \tilde{y} - \tilde{x}M, \quad \tilde{x} = \frac{\sum_i \alpha_i x_i}{\alpha}, \quad \tilde{y} = \frac{\sum_i \alpha_i y_i}{\alpha} \tag{3.3}$$

where $\alpha = \sum_i \alpha_i$ and $\hat{x}_i = x_i - \tilde{x}$ and $\hat{y}_i = y_i - \tilde{y}$.

Using the above, we can rewrite the minimzation of Eqn. 3.1) as

$$\sum_i \alpha_i \|\hat{x}_i M - \hat{y}_i\|^2, \quad \text{subject to } M^T M = I \tag{3.4}$$

which can next be represented in matrix form as

$$\|AM - B\|_F, \quad \text{subject to } M^T M = I \tag{3.5}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, and matrices A and B are given by

$$A = \begin{bmatrix} \sqrt{\alpha_1}\hat{x}_1 \\ \sqrt{\alpha_2}\hat{x}_2 \\ \vdots \\ \sqrt{\alpha_k}\hat{x}_k \end{bmatrix}, \quad B = \begin{bmatrix} \sqrt{\alpha_1}\hat{y}_1 \\ \sqrt{\alpha_2}\hat{y}_2 \\ \vdots \\ \sqrt{\alpha_k}\hat{y}_k \end{bmatrix} \tag{3.6}$$

Minimizing the function in Eqn. 3.5) with respect to M is a known example of the orthogonal Procrustes Problem [78]. The solution of this problem is known, and given by

$$M = UV, \quad A^T B = UDV \tag{3.7}$$

where UDV is the singular value decomposition (SVD) of $A^T B$.

Having M delivered by the SVD process, we now can compute the desired projection y of a regular point x simply by substituting the expression of M (from Eqn. 3.7) and t (from Eqn. 3.3), to obtain

$$y = f_x(x) = (x - \tilde{x})M + \tilde{y} \tag{3.8}$$

The reader may, at this point, object that computing a SVD dcomposition for each data point x from a dataset having potentially tens of thousands (or more) uf such instances is simply computationally prohibitive, especially in the context of an interactive application. Yet, the matrix $A^T B$ which we need to decompose by SVD to compute M (Eqn. 5.1) is an $m \times 2$ matrix (has only two columns). As such, computing an SVD for this matrix can be done extremely efficiently using high-quality numerical packages such as [21]. Specifically, such a decomposition, and therefore the projection of an instance x, is $O(k)$ for k control points. As such, the entire LAMP projection is $O(kn)$ for n regular points. Formally speaking, to keep a good local control of a projection, we need to have more control points when the number of regular points increases, or $k = \phi n$, where ϕ can be a small percentage. As such, the complexity of LAMP is $O(n^2)$. However, in practice, since f is very small, LAMP's actual run-times are far below quadratic in the number of regular points. We detail the implications of the LAMP minimization scheme described above further in terms of accuracy and computational speed in Section 3.4.

3.3.2 *Control Point Analysis*

As outlined previously, the quality and flexibility (degree of local control) of LAMP are both influenced by the choice of control points. This choice is further discussed below.

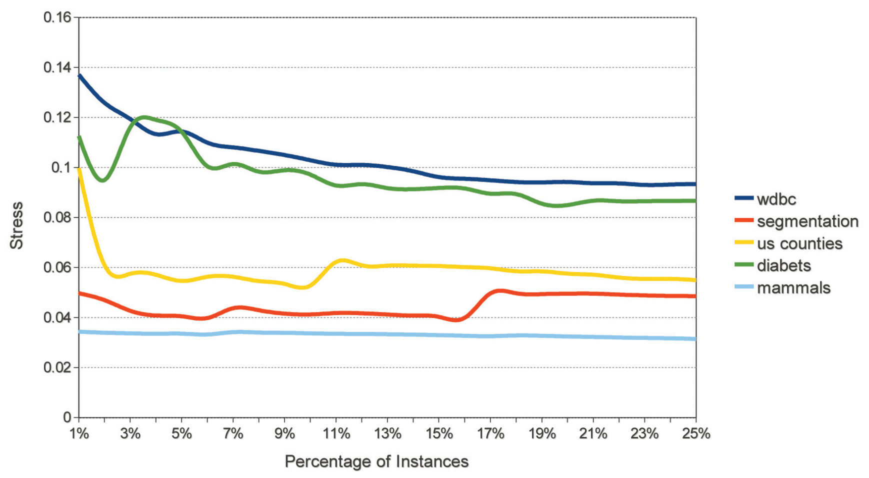


Figure 3.3: The stress quality metric of LAMP as a function of the number of control points. These range from 1% to 25% of the total point count. Experiments done on five dataset (described separately in Table 3.1 for details about the data sets).

Number of control points: The first aspect we study is how many control points we chose from the total number of points to project, or the value of $\phi = |\mathcal{X}_S|/|\mathcal{X}|$. Let us note here that existing control-point-based methods such as PLMP [164], Pekalska's [171] and PLP [167] have strong limitations on the minimum number of control points they need to use. In detail, PLMP and Pekalska require a number of control points (at least) equal to the dimensionality m of the data; and PLP requires a minimum number of control points in each local neighborhood graph (which, in practice, means that ϕ has to be a quite large fraction of $|\mathcal{X}|$). In practice, the above methods advise using $k = \sqrt{|\mathcal{X}|}$ control points, which makes $\phi = 1/\sqrt{\mathcal{X}|}$.

In contrast, LAMP has far such limitations with respect to ϕ. To show this, we ran the method on five datasets, using values of ϕ ranging between 1% and 25% of $|\mathcal{X}|$. Figure 3.3 shows the results. To evaluate the quality of the projection, we compute the aggregated normalized stress function $\frac{\sum_{ij}(d_{ij}-\bar{d}_{ij})^2}{\sum_{ij} d_{ij}^2}$ (d and $\bar{d}$ are the distance between instances p_i and p_j in $\mathbb{R}^m$ and $\mathbb{R}^2$ respectively). The key observation here is that the stress metric does not increase considerably when the number of control points decreases, which tells us that LAMP can create good-quality projections (from a distance preservation perspective) even with a fraction ϕ of control points much smaller than the $\phi = 1/\sqrt{\mathcal{X}|}$ advertised by PLMP, PLP, and Pekalska.

Choice of control points: A second aspect is which of the regular points $\mathcal{X}$ should one select to use as control points. This is a much harder problem to answer in general, as it depends on which particular aspects of the dataset $\mathcal{X}$ does one want to preserve better (in favor of other aspects) in the projection. For instance, if one knows *a priori* that certain neighborhoods in $\mathbb{R}^m$ are more important, for the problem at hand, than others, one should sample them with more control points. Alternatively, if one desires to next manipulate specific neighborhoods on $\mathbb{R}^m$ in the final interactive projection, these neighborhoods should also be sampled by sufficiently many control points. Without any information on the nature of the data and type of problem to solve, a simple strategy is to select k control points in $\mathbb{R}^m$ based on a clustering or data-density-analysis process (*e.g.*, using mean shift [44]), so as to ensure that every neighborhood in $\mathbb{R}^m$ is fairly in the vicinity of a control point. As this issue is, as explained, strongly data and task dependent, we will not explore it further to provide a generic solution.

Placement of control points: A third aspect relates to how control points are placed in $\mathbb{R}^2$. Two sub-issues exist here. First, some initial placement is required. For this, we use an accurate force-based scheme [214] to place randomly selected control points in $\mathbb{R}^2$. The use of the force-based scheme is here computationally motivated because, as explained, we use a quite small number of control points. As such, this step does not affect LAMP's performance. Secondly, given such an initial placement, users can next manipulate desired control points to reorganize the projection. This issue is discussed separately in Section 3.5.

Influence of control points: A final issue to discuss is how strongly do control points influence the regular points in the projection. As outlined already, the number of terms in the summation in Eqn. 3.4 can be varied to modify the projection's behavior. Using more terms creates a 'stiffer' projection which has less discontinuities (thus, less points that exhibit large projection errors), but is harder to locally adapt next in terms moving the control points. Conversely, using less terms creates a 'looser' projection which offers more flexibility in local adaptation, but can exhibit a large number of discontinuities. Both above scenarios have their advantages and disadvantages, depending on the specific application at hand. These issues are discussed in the next section.

3.4 RESULTS AND COMPARISONS

The following results were generated by running LAMP on an Intel Core i7 CPU at 2.66GHz with an NVIDIA® Quadro FX 3800 video card with 8GB of RAM memory. LAMP was coded in Java using the numerical library JBlas [25] to compute the SVD decomposition.

To demonstrate and confirm the proposed projection's quality, we present two different analysis and comparisons. The first aims to evaluate the performance

Table 3.1: Data sets used in the comparisons.

Name	Size (number of instances)	Dimensions	Source
Wdbc	569	30	[71]
Diabetes	768	8	[71]
Segmentation	2,100	19	[71]
US counties	3,028	14	[87]
Isolet	6,238	617	[71]
Letter rcn	20,000	16	[71]
Mammals	50,000	72	[71]
Viscontest	200,000	10	[238]

in terms of accuracy and computation time, while in the second we use quantitative measures applied after control point manipulation.

To assess the first case, we compare nine existing multidimensional projection (MP) techniques against LAMP employing eight data sets with comprehensive variation of size (number of instances) and dimensionality (Tab. 3.1). We use the following criteria to choose the MP techniques employed in the comparisons:

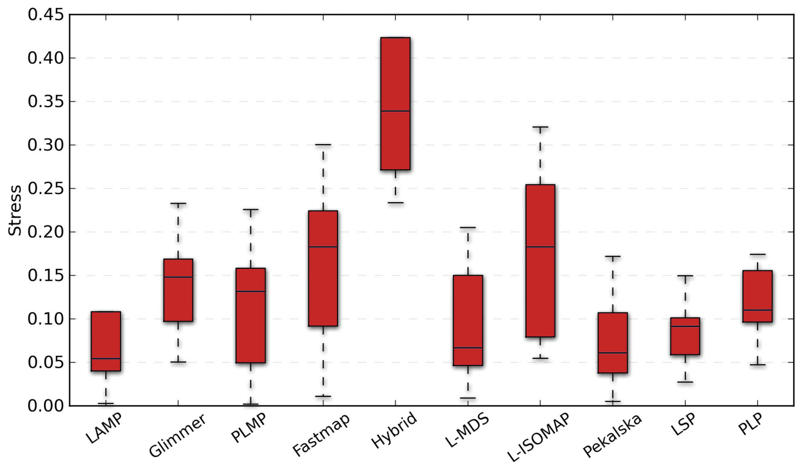
Criterion a): They have a good performance by means of stress and/or computation time, which allows the comparison of LAMP against efficient techniques, and

Criterion b): They integrate user intervention and manipulation when building the projection, which is done by using a subset of samples (control points) to carry out the mapping.

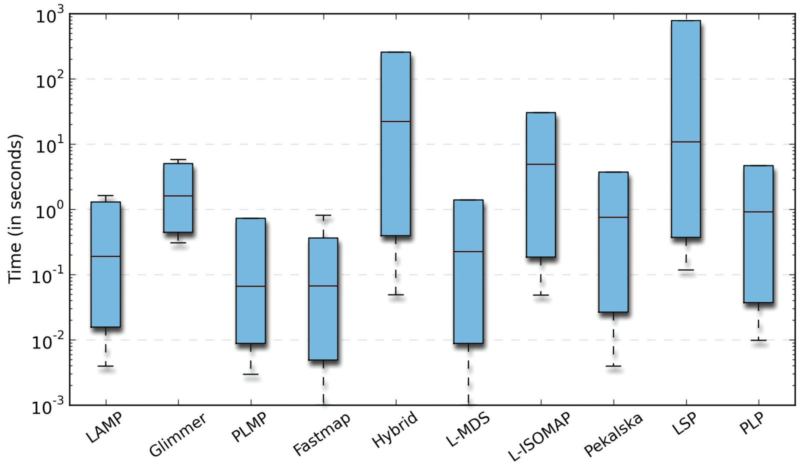
The selected techniques we chose for the comparison that handle both criteria are: LSP [163], PLP [167], PLMP [164], Hybrid [106], L-MDS [54], Pekalska [171] and L-Isomap [205] since all have good stress and computational time results, and they also use a subset of samples in the construction of the MP. Moreover, we also selected Glimmer [98] due to its good stress performance and Fastmap [69], known as a fast projection method.

Regarding the accuracy comparison, the boxplots in Figure 3.4a show that LAMP is one of the most accurate techniques, being comparable to Pekalska, which is a highly precise method. In terms of computational time, Figure 3.4b shows that LAMP is only slower than PLMP and Fastmap, which both are techniques well known for their very low computational cost. However, LAMP is quite competitive when comparing to state-of-art methods such as PLP.

Another visual tool to assess LAMP's accuracy are the scatterplots that map *original-distance* $\times$ *projected-distance*, illustrated in Figure 3.5, where top-left numbers correspond to normalized stress and computational time (seconds). We can notice that LAMP yields scatterplots that look very close to thin bands oriented at 45 degrees in almost all test cases – with the ideal representation being a



(a)



(b)

Figure 3.4: Boxplots comparing stress and computational times of ten techniques, including LAMP.

diagonal line. This means that original distances are well preserved when projected in the visual space. This did not occur for other MP techniques such as L-Isomap and Hybrid, which show a more spread distribution of points in the scatterplots, clearly visible in mammals data set, for example.

The first set of comparisons discussed above correspond confirms the accuracy and computational efficiency of LAMP. However, as LAMP lets dynamically interfere in the projection via control points, it is desirable to analyze the LAMP's effectiveness when producing such mappings. For this second round of comparisons, we outlined two quantitative measures that evaluate the mappings

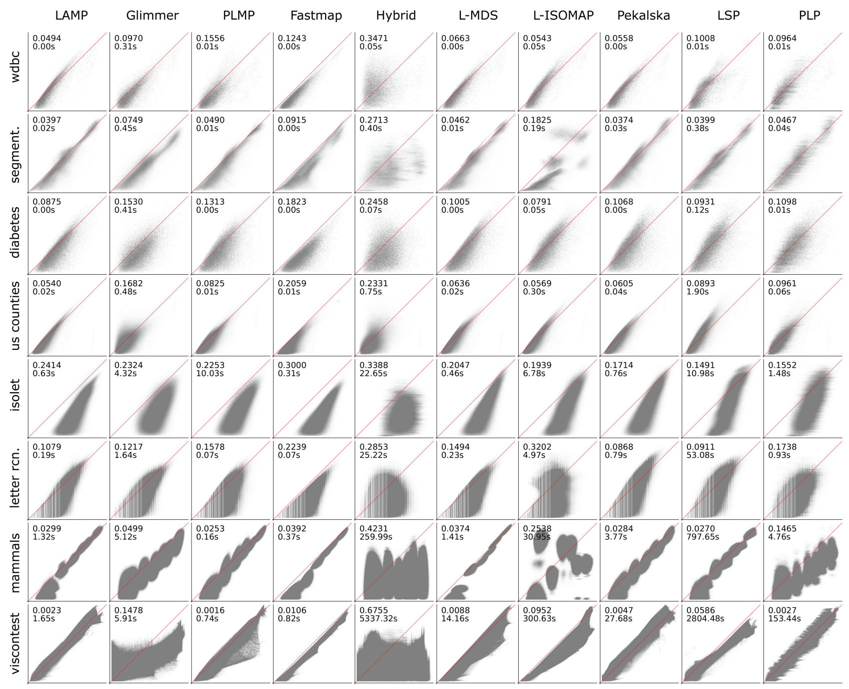
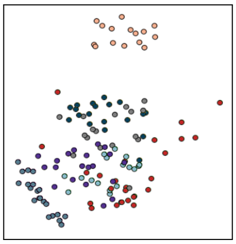


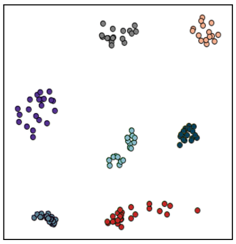
Figure 3.5: Scatterplots *original-distance* $\times$ *projected-distance* of ten techniques, including LAMP.

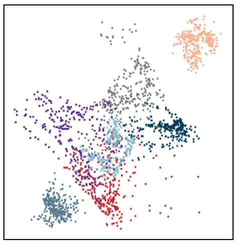
after interactive control points manipulation, namely, *neighborhood preservation* and *silhouette coefficient* (described previously in Sec. 2.4.3).

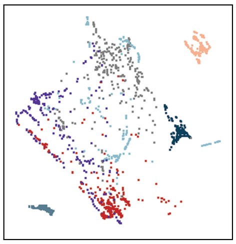
Figure 3.6 presents the effects of user manipulation of control points for the Segmentation data set. While Figure 3.6a presents the force-based control points mapping, Figure 3.6b shows the result of a better grouping of control points in the visual space after user manipulation. The LAMP, LSP, Pekalska and PLMP methods produce, respectively, the mapping of Figures 3.6c to 3.6f, with all of them using the control-points layout in Figure 3.6b. To calculate the affine map f_x for each instance x, LAMP used all control points. Regarding the $\mathtt{Silh}$ coefficient, LAMP is better than PLMP but as not as good as the ones produced by Pekalska and LSP. However, in Figures 3.7a to 3.7d we can clearly see that the situation changes when LAMP makes use of the nearest control points. When LAMP used 75%, 50%, 25%, and 10% of the nearest control points to build the mappings, the $\mathtt{Silh}$ coefficient increases consistently and reaches a higher silhouette value than the global methods LSP, PLMP and Pekalska.

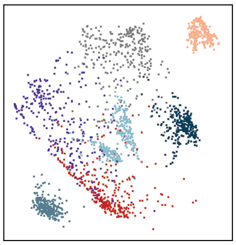
Measures involving neighborhoods can also be defined from information computed in the visual (2D) space. An example is the PLP method that is based on distances computed in the visual space to build neighborhood graphs which derive the projection maps. Paulovich *et al.* [167] demonstrate the use of neighborhoods information leverages the capability of steering the projection consid-


(a) Force-based control points mapping


(b) User provided layout


(c) LAMP Silh = 0.4003


(d) LSP Silh = 0.4584


(e) Pekalska Silh = 0.5083

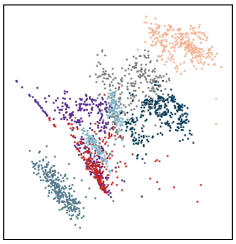
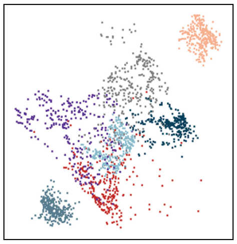
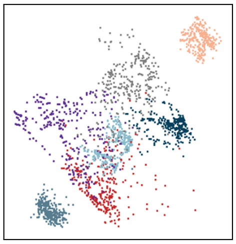
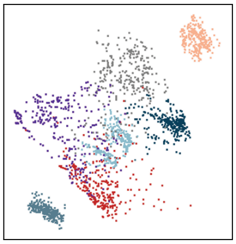

(f) PLMP Silh = 0.2475

Figure 3.6: LAMP, LSP, Pekalska and PLMP projections when the user manipulated the control points.


(a) 75% Silh = 0.4166


(b) 50% Silh = 0.4365


(c) 25% Silh = 0.4682

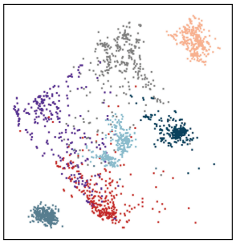
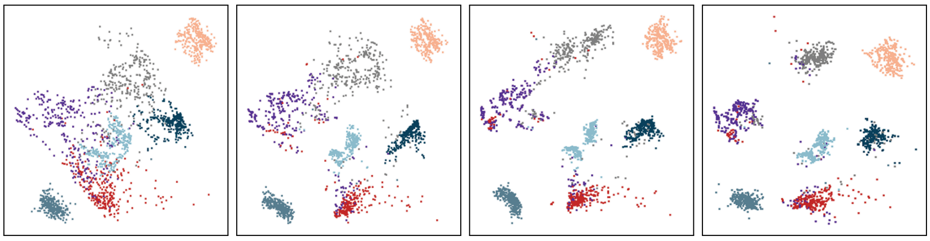

(d) 5% Silh = 0.5235

Figure 3.7: LAMP projection when varying the percentage of nearest control points.



(a) 75% $\mathtt{Silh} = 0.4590$ (b) 50% $\mathtt{Silh} = 0.5496$ (c) 25% $\mathtt{Silh} = 0.5570$ (d) 5% $\mathtt{Silh} = 0.5644$

Figure 3.8: LAMP projections from neighborhoods computed in the visual space. From left to right, the result of using 75% to 5% percent of the nearest control points.

ering to the control point positions. Those results encouraged us to adapt LAMP, taking into account 2D information when constructing the affine mappings. If we consider an instance x in $\mathbb{R}^m$, where x_i is the control point nearest to x, we can modify LAMP to use control points x_j whose images y_j are closer to the image y_i of x_i in the visual space, instead of use control points in the neighborhood of x. As a result, the mappings that use 2D neighborhood information push the projection of x toward the control points used in the computation of $f_x(x)$, meaning that they may not necessarily be neighbors of x in the original space.

LAMP becomes highly sensitive to the control points' positions in the visual space due to the use of 2D information. This produces mappings that follow the (2D) layout of the control points very closely. Figures 3.8a to 3.8d confirm this statement by showing mapped instances that becomes progressively more grouped when the percentage of nearest neighbors ranges from 75% to 5% and neighbors are defined based on the visual space. Another advantage to use 2D information can measured by the silhouette coefficient that confirms a good group-preservation of LAMP, better than PLP (Figure 3.9). Figure 3.10 also shows LAMP's superiority considering neighborhood-preservation graphs, preserving more than fifty percent of neighbors, on average.

Finally, we show LAMP's robustness when dealing with a reduced number of control points. This is an important property, as our applications discussed next in Sec. 3.5 all require manipulating a limited number of such points. Three control points were randomly selected for each one of the seven classes that composes the Segmentation data set (Figure 3.11a), and interactively placed in the visual space to keep the classes as much separated as possible. The resulting mapping of LAMP with a reduced set of control points is shown in Figure 3.11b, with neighborhoods defined from the visual space. It is important to notice two important aspects: *i)* even with a reduced control points (21 in total), LAMP projected the data set consistently, obtaining a silhouette coefficient value comparable to the one produce by PLP with much more control points (see Figure 3.9); *ii)* Pekalska, PLMP and PLP cannot build mappings using such a small amount of

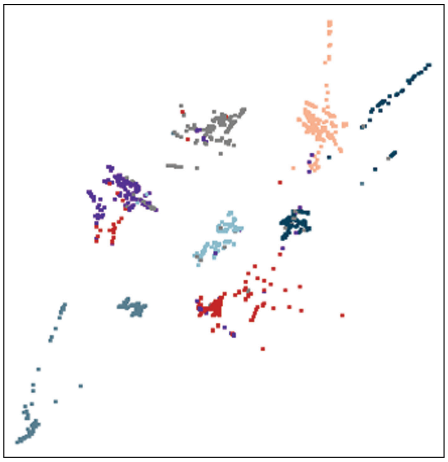


Figure 3.9: PLP projection using 2D distances (visual space) with the control points' layout shown in Figure 3.6b ($\mathtt{Silh} = 0.4411$).

control points; *iii)* Pekalska and PLMP cannot project a strong separation such as the one produced by LAMP in Figure 3.8d, due to their global nature.

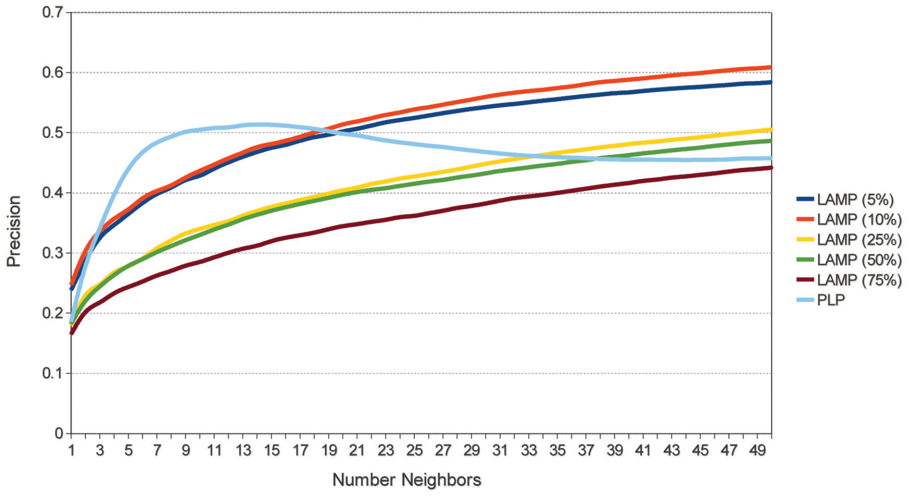


Figure 3.10: LAMP and PLP neighborhood preservation.

3.5 APPLICATIONS

Incorporating user knowledge into the mapping process by interacting with projected data is a useful property in the sense that can be explored in many

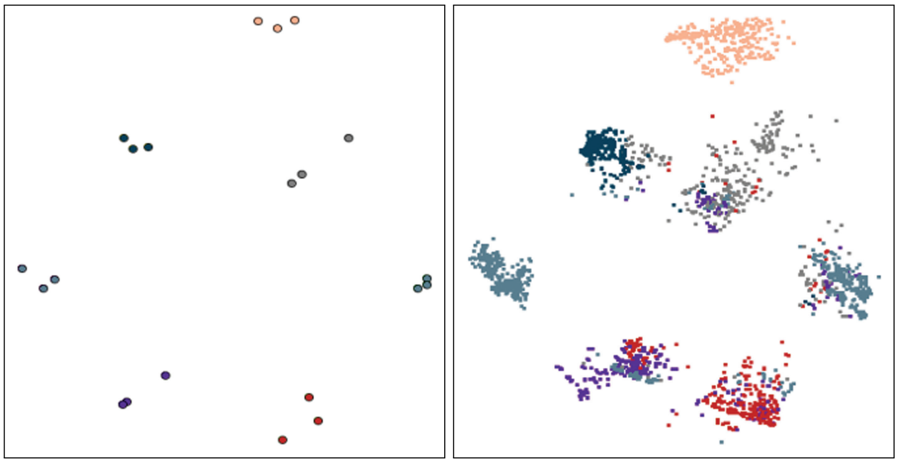

(a) 3 control points for each class. (b) LAMP Projection ($\mathtt{Silh} = 0.4361$)

Figure 3.11: LAMP Projection (neighborhood in R^2) using only 3 control points per class (21 in total, against 137 needed to execute PLP).

visualization problems. As an example, we next present two applications that take advantage of LAMP's flexibility and robustness.

3.5.1 *Application 1: Correlating images and audio data*

Dealing with heterogeneous data sets is not an easy task, especially if they are derived from different sources. In such cases, it is quite useful to work with user-assisted data correlation applications that aim to relate instances from data sets that do not have any connection. The key idea here is to start with a reduced number of control points selected from the unrelated data sets, and let the user freely manipulate these control points in the visual space, bringing closer instances that the user wants to correlate. Once all control points from distinct data sets have been correlated, meaning that they are grouped as desired in the visual space, the LAMP method can be used to project all the remaining instances from each data set. As LAMP builds mappings that follow the layout of the control points, instances from the distinct data sets that are projected closer to each other in the visual space are *expected* to be correlated

Based on the visual data-correlation framework described above, we developed a system that correlates images and music. The main idea is to automatically create slide-shows with sound by associating certain genres of music to specific kinds of pictures. A step-by-step illustration of our framework is presented in Figure 3.12. Thereby, the framework applied to correlate music and images extracted from videos operates as follows:

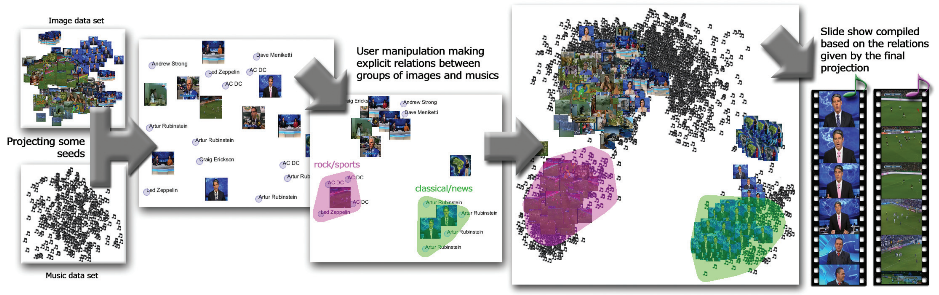


Image data set
Projecting some seeds
Music data set
User manipulation making explicit relations between groups of images and musics
rock/sports
classical/news
Slide show compiled based on the relations given by the final projection

Figure 3.12: Description of an interactive systems that uses LAMP to correlate different data sets that do not have explicit relation among instances.

1. The user starts by choosing music from different genres, such as a couple of classic music and a few rock-n-roll tracks.
2. Next, the user selects a few pictures belonging to distinct classes, such as pictures of cars, aircrafts, and houses from a set of images. The user can also explicitly select images and music from other classes to represent instances that should not be correlated.
3. An initial projection is created using a few instances of music and images for the user, who next interacts by selecting them in order to bring them as close as possible in the visual space. For example, one can group houses and classic music in the first group; and cars and aircrafts images, and rock-n-roll tracks, in the second group. This step creates the explicit correlation between the unrelated instances.
4. Finally, LAMP considers the above manipulation to map the remaining pictures close to their best-fitting tracks. For instance, house pictures will be placed in the same neighborhood where classic music hits are projected. Next, images and data are fused: In our example, house pictures are used to form a slide show whose soundtrack is automatically composed by the classic music mapped in their neighborhood. In other words, the produced slide shows contain images and music playing in a synchronized manner.

We have developed a prototype system to accomplish the tasks of corresponding image and music mentioned before (see Figure 3.13). The accompanying video [104] shows how image and sound can be easy correlated using that system, providing a better idea of how LAMP works and its effectiveness when integrated in the proposed application.

Next, we explain in detail each interaction step of the prototype framework shown in Figure 3.13.

1. First, a reduced number of instances representing the control points are selected for both music and pictures data sets (top left).

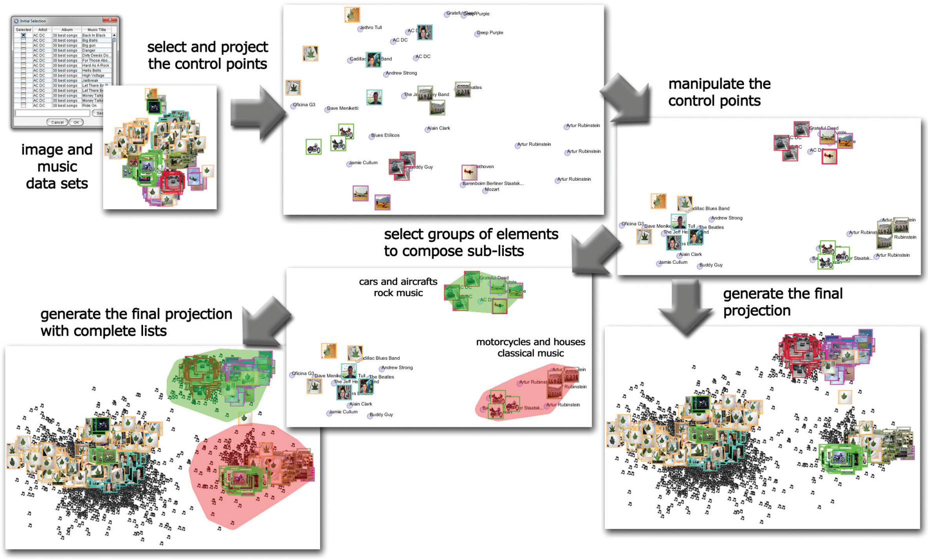


image and music data sets
select and project the control points
manipulate the control points
select groups of elements to compose sub-lists
generate the final projection
generate the final projection with complete lists
cars and aircrafts rock music
motorcycles and houses classical music

Figure 3.13: Prototype system correlating image and music.

2. These control points are placed/projected in the visual space (top middle). Image thumbnails represent the pictures, while purple circles labeled with singer/group band names represent the musics.

3. Next, user can freely interact by selecting and picking both samples of pictures and musics, grouping the ones that must be correlated (top right). In our framework scenario was created 3 groups. At this step, we can choose between generating a preview of the final projection (step 4) or directly select the groups to compose the slide show with sound (step 5).

4. If the user choose the first option above, the framework allows the user to check if the final projection has a good layout, by mapping the remaining instances based on the previous interaction (bottom right). In case of not having an acceptable layout, the user can redo the interaction process (step 3).

5. The user can brush multiple regions of control points in the visual space (bottom middle), selecting groups of elements to compose sub-lists, illustrated by the two groups in the bottom middle figure: 1) cars and aircrafts pictures associated with rock music, and 2) motorcycles and houses pictures associated with classical music.

6. In the final step (bottom left), is generated the final projection with the complete lists, mapping all the remaining pictures and music, making up one slide show with sound for each brushed group (or lists).

To develop and test our multimodal framework, we have proceeded as follows. For the image part we used the *Caltech database* that contains 3,100 pictures [70]. A total of 150 image features were extracted by employing the *bag-of-visual features (BoVF)* [243] technique. Regarding the audio media, together with a database with 3,857 music tracks we used the *JAudio Tool* [74] to obtain the low-level features from `mp3` files. To compound the result vector with 78 dimensions, we extracted features such as beat points, statistical summaries, and so on.

The system mentioned above that correlates different media sources is only a proof-of-concept of a new visualization-based paradigm for correlating distinct data sets. In fact, very little has been done in the direction of developing visual mechanisms to correlate distinct data sets. Thus, LAMP and its good properties contribute to bring out new perspectives to this type of application.

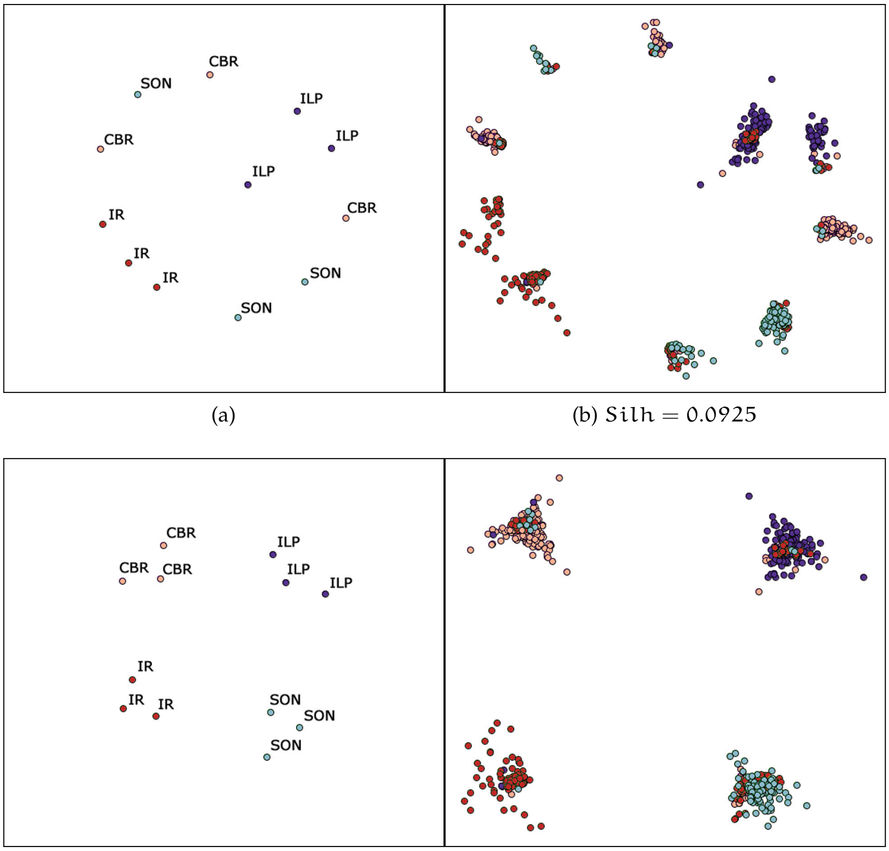
3.5.2 *Application 2: Document analysis*

Document analysis is another type of application where LAMP can play an important role. Usually, textual documents form a high dimensional space, making distance metrics like Euclidean and cosine less able to discriminate text documents. Hence, user knowledge and user interaction are very important to organize and group documents according to their similarity. However, this is not a trivial task for the user because the one has to *i)* study the summary (set of key words), which is responsible to describe each document, and also *ii)* to group the ones whose key words match closely. Besides, MP methods such as PLP and PLMP require a large amount of control points in the projection process, which is unfeasible in this context.

In contrast, LAMP can produce projections with a reduced number of control points, allowing users to manipulate few documents to get good projection results. Figure 3.14 illustrates an application of document collection projected by using LAMP, that contains 675 scientific papers from four distinct areas: Information Retrieval (IR), Inductive Logic Programming (ILP), Sonification (SON), and Case-Based Reasoning (CBR).

Considering each document a high-dimensional set of attributes, we apply the vector-space-model approach [190] to extract the frequency of relevant terms from abstract, title, references, and authors of each document. Then, the attributes of each document correspond to the most-frequently encountered terms over the entire document set under study, describing precisely the frequencies of the above terms. In our scenario, we got 390 attributes per document, or in other words, a 390-dimensional dataset.

The text documents application scenario is shown in Figure 3.14. We picked out from the document dataset 3 control points from each document class, totaling 12 (Fig. 3.14a). We next placed these 12 control points in the visual space using a force-based scheme [214]. In the projection of Figure 3.14b, we can see that documents from the same class are not properly grouped together, resulting in a tangled mapping. Once LAMP supports a small number of control points to manipulate, it is an easy task to identify similar textual instances by grouping them in the visual space (Fig. 3.14c). The last step, a final projection is created by using LAMP with the previous control points, ensuring a good preservation provided by the user layout, as can be seen by the silhouette coefficient in Figure 3.14d). We can also notice in this figure that colors are used to highlight documents belonging to the same class, while this class information is *not used* by the system.



(a) (b) $\mathrm{Silh} = 0.0925$

(c) (d) $\mathrm{Silh} = 0.4207$

Figure 3.14: Application exploration of text documents.

3.6 DISCUSSION

The qualitative and quantitative comparisons presented in Section 3.4 clearly show the effectiveness of the LAMP technique, which outperforms state-of-art methods with respect to properties such as accuracy, robustness, flexibility, and ease of use. The solid mathematical foundation of LAMP ensures a better performance and distance preservation, while in the same time it allows us to incorporate user knowledge in the projection process. Another advantage of LAMP is the method's simplicity – we basically only require an implementation of a SVD matrix-decomposition.

Many application scenarios can take advantage of the fact that LAMP uses few control points to accurately map instances and lets the user freely interact with the position of control points in the visual space. We have shown that LAMP can be easily applied in application scenarios such as visual exploration of text documents as well as the novel visualization-based multimedia-data correlation tool described in the previous section. We believe that the multimedia-data correlation framework has a high potential, being easily adapted to work with datasets from other domains such as social networks and scientific data.

"Pleasant" layouts were obtained when 1) control points x_i and their image y_i are in the same scale, or 2) when we normalize data in the original space as well as the control points position in the visual space. The exact amount of neighbors to create a good layout that keep similar instances grouped together is an aspect that requires further studying. An alternative to the k-nearest neighbors technique used in our approach would be to use a so-called range search. However, the exact radius (range) setting used for each control point needs further study.

3.7 CONCLUSION

In this chapter, we have proposed a projection technique called Local Affine Multidimensional Projection (LAMP) for the creation of two-dimensional projections of high-dimensional datasets. In comparison to existing multidimensional projection techniques, LAMP combines a number of important advantages: (a) it has a solid mathematical foundation, ensuring robustness and versatility; (b) it is very computationally efficient, allowing it to create projections of large datasets at interactive rates; (c) it allows the detailed control of the resulting two-dimensional layout by the interactive manipulation of a small number of data instances. The quantitative and qualitative comparisons shows that LAMP outperforms existing state-of-the-art projection techniques, not only in terms of stress minimization but also in terms of very competitive computational time.

The combination of above features allows LAMP to support the construction of interactive techniques that support the visual correlation of several high-dimensional datasets. We have shown how this feature is instrumental in constructing applications that enable the easy exploration and organization of multimedia-related datasets such as images and music and text document collections, in ways which could not be efficiently addressed until now. Given the above, LAMP will be the favored projection technique to be used in the next chapters, both for studying and evaluating novel explanatory tools for multidimensional projections, and also for exploring multidimensional multimedia data.

4 VISUAL ANALYSIS OF DIMENSIONALITY REDUCTION QUALITY FOR PARAMETERIZED PROJECTIONS

ABSTRACT: *Many dimensionality reduction (DR) algorithms have been proposed for visual analysis of multidimensional data. Given a set of n-dimensional data points, such methods create a low-dimensional projection that preserves relative distances and/or neighborhoods. However, the overall quality and usability of the projections being produced depends on many factors, like the DR techniques used and their several parameters. As such, users are challenged in assessing the suitability of a projection for specific tasks, in finding how they can tune the projection method's parameters to obtain desired results, and also in comparing different projections. We propose several interactive visualizations to assist the above tasks. These visualizations depict the quality of a projection and also show how this quality related to parameter choices of the projection algorithms. Quality is modeled in terms of distance preservation, by emphasizing false and missing point-neighbors. Parameter settings are explored by showing how the projection quality depends on these. Our visualizations are scalable to large and dense projections by using several image-based techniques. We demonstrate our techniques using several recent DR techniques and several multidimensional datasets, and show how insight in projection quality and related parameter settings can be easily obtained, without burdening users with the internal details of the DR algorithms being used*[1].

4.1 INTRODUCTION

The previous chapter has described LAMP, a new technique for dimensionality reduction (DR) that exhibits attractive properties in terms of scalability, ease of use, level of local control, robustness, and accuracy. As we have seen, LAMP achieves better global stress values than other comparable state-of-the-art techniques. However, as any other projection technique, LAMP will inherently cause various forms of projection error. Depending on the application context (problem at hand, datasets, types of end users involved), these errors may be essential to be discovered, or alternatively they may not influence the overall usage and interpretation of the projection results. Additionally, as LAMP allows local control of the projection (by its control points), additional degrees of freedom exist by which users can influence the final projection outcome, and thus its quality. As such, a central question emerges: How can we present existing projection errors, or more generally how can we present the *quality* of a projection, to its end

1 This chapter is based on the paper *Visual Analysis of Dimensionality Reduction Quality for Parameterized Projections* (R. Martins, D. Coimbra, R. Minghim, A. Telea), Computers and Graphics, vol. 41, pp. 26-42, 2014. The first two authors had equal, and major, contributions to this paper,. and should as such be seen as joint first authors. Specific contributions of D. Coimbra involve: the proposal of the various error-encoding techniques (Sec. 4.4); the proposal of the visual analytics workflow (Sec. 4.4.8); and the selection of datasets, projections, and projection parameters, and the corresponding comparison and interpretation of projection techniques in Sec. 4.5.

users, so they can immediately see where potential problems occur, and how these may influence their interpretation and usage of the projection?

In this chapter, we present a set of visualization techniques that help users with exploring the link between DR algorithm parameter settings and the quality of the resulting projections. Our visualizations target the following questions:

- How does the projection error affect different regions of a 2D projection?
- How to find so-called missing neighbors (points which are projected too far apart with respect to their nD distance)?
- How to find so-called false neighbors (points which are projected too close with respect to their nD distance)?
- How do the above quality aspects compare for different DR methods and their parameter settings?

For this, we propose 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 DR algorithm. We illustrate our visualizations by exploring the parameters of five state-of-the-art DR techniques for several real-world datasets.

This chapter is structured as follows. Section 4.2 presents related work on DR algorithm quality analysis. Section 4.3 presents our analysis goals. Section 4.4 describes our proposed visualizations. Section 4.5 uses these methods to explore the quality, as function of DR method parameters, of several DR techniques. Section 4.6 discusses our results. Section 4.7 concludes the paper.

4.2 RELATED WORK

As outlined in Sec. 2.4.3, a large class of multidimensional projection methods aim to map high-dimensional data to a m-dimensional visual space, where $m = \in \{2, 3\}$, so as to preserve distances as much as possible. Implicitly, if distances are well preserved, so are neighborhoods too. Other projection methods aim to optimize neighborhood preservation explicitly, with less concern for a faithful preservation of distances [225, 227]. Our focus here is mainly on projections which achieve neighborhood preservation mainly by preserving distanced, such as *e.g.* LAMP, discussed in Chapter 3.

Considering a dataset $D^n = \{\mathbf{p}_i \in \mathbb{R}^n\}_{1 \leqslant i \leqslant N}$ of N n-dimensional points, dimensionality reduction (DR) can be seen as a function

$$f : \mathbb{R}^n \times P \to \mathbb{R}^m \tag{4.1}$$

which 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. P denotes the *parameter space* of f, *i.e.* the various settings that control the projection algorithm, including the algorithm type itself. f is designed to keep the so-called *structure* of the data as similar as possible in $\mathbb{R}^n$ and $\mathbb{R}^m$. In our case, as explained earlier, we focus here on preserving the data structure implied in terms of distances between points.

Among particularities of several multidimensional projection methods, a very important topic is how good/bad the projection reflects the original data. As discussed in Sec. 2.4.3, there are several quantitative measures to assess the quality of projection, however how the literature visualizes these projection quality? Several techniques are present in the literature to this end, as follows.

Although projection quality is acknowledged as important, most DR literature considers mainly aggregated quality metrics such as the stress function (Eqn. 2.6), correlation [242], neighborhood preservation average plots [163], and distance scatterplots [103], which are distance and neighborhood based metrics, and cluster segregation metrics [200]. 2D scatterplots can show the correlation of D^n with D^m [103]. Such metrics capture the *overall* quality of a projection, but do not help finding local quality variations. In other words, they do not show projection problems for *any* point i *vs all* points $j \neq i$ in the input dataset. Such fine-grained insight is useful in cases where users want to reason about specific small-scale errors in a projection, *e.g.*, find out if specific subsets of the projected points are indeed placed correctly with respect to other specific subsets of points.

Local metrics can be used to highlight where (in a projection) errors happen. For this, Schreck *et al.* compute, for each $\mathbf{p} \in D^n$, the projection precision score (*pps*) defined as the normalized distance between the two k-dimensional vectors having as components the Euclidean distances between $\mathbf{p}$ and its k nearest neighbors in D^n, respectively D^2 [197]. Visualizing *pps* as a color map shows areas where neighborhoods are not preserved. However, a neighborhood cannot be preserved for two distinct reasons: true neighbors (in D^n) are missing (in D^2), or neighbors (in D^2) are actually false neighbors (in D^n). The *pps* metric does not differentiate between such situations, and can also be sensitive to permutations of points that do not change distances.

Recognizing that DR methods can create distance approximation errors, Van der Maaten *et al.* extend the t-SNE technique [225] to output a set $\{M_i\}$ of 2D projections rather than a single one [226]. 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 [10]. 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 measures 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. Our proposed techniques in Secs. 4.4.2, 4.4.3, and 4.4.4 adapt and extend these visualizations in several directions.

Still using colored Voronoi cells, Lespinats and Aupetit show, at the same time, point stretching and compression by using a 2D color map [127]. The proposed color map encodes stretching as green, compression as purple, low-error points as white, and points with high stretching (also called *tearing*) 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 which 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.

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 $\mathbb{R}^2$ [89]. 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 [10, 127], points are colored using Shepard interpolation, which yields a smoother, and arguably less distracting, image. However, in contrast to [10, 127], this method can only show errors related to a selected focus point.

4.3 EXPLANATION GOALS

In general, a projection technique f should preserve the *structure* of the original space $\mathbf{R}^n$. As outlined earlier, this can be seen as a mix of distance and neighborhood preservations taking place at different scales – for instance, k-nearest neighborhoods can be preserved up to some given value of k; or distances can be preserved up to some given value d_{max}. For users, the projection's *precision* [197] is not clear unless they can interpret projected neighborhoods adequately [10]. Thus, given any DR algorithm (Eqn. 4.1), 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 in 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:

A. False neighbors: Consider a point $\mathbf{p}_i \in D^n$ and its counterpart in the projection $\mathbf{q}_i = f(\mathbf{p}_i)$. Neighborhood preservation implies that *all* points $\mathbf{q}_j$ which are close to $\mathbf{q}_i$ (in 2D) should be projections of points $\mathbf{p}_j$ which are close to $\mathbf{p}_i$ (in D^n). When this does not happen, *i.e.* we have a $\mathbf{q}_j$ close to $\mathbf{q}_i$ for which $\mathbf{p}_j$ is not close to $\mathbf{p}_i$, one can erroneously deduce from the projection that $\mathbf{p}_j$ is close to $\mathbf{p}_i$. We call such a point j a *false neighbor* of i.

B. Missing neighbors: Apart from the above, neighborhood preservation implies that *all* $\mathbf{p}_j$ which are close to $\mathbf{p}_i$ (in D^n) project to points $\mathbf{q}_j$ which are close to $\mathbf{q}_i$ (in 2D). When this does not happen, *i.e.* we have a $\mathbf{p}_j$ close to $\mathbf{p}_i$ for which $\mathbf{q}_j$ is not close to $\mathbf{q}_i$, one will not be able to visually find the complete set of points similar to point i. Points j that are missed in this search are called *missing neighbors* of i.

C. Groups: Projections are frequently used to visually find groups of similar points, *e.g.* topics in a document collection [103, 163] or classes of strongly-similar images in an image database [69]. As such, the notions of false and missing point-neighbors can be extrapolated, for groups, to *false members* and *missing members* respectively. That is, given a group Γ of closely-projected points, points visually located in Γ that do not truly belong there are called false members; and points visually located outside Γ that actually are strongly similar to points in Γ are called missing members, respectively.

D. Completeness: As explained earlier, aggregated local metrics such as those in [197, 10, 127, 89] show, up to various extents, where missing or false neighbors occur. However, they do not fully and explicitly highlight the identities of all those neighbors for each projected point; and also do not cover the case of false and missing group members.

4.4 PROPOSED VISUALIZATIONS

We next introduce several visualization techniques that target the analysis goals stated in Sec. 4.3. As a running example, we use LAMP as projection method, with the default parameter settings given in [103] and discussed separately in Chapter 3. As test dataset, we use the well-known 19-dimensional Segmentation dataset with 2300 observation, which was previously used in several dimensionality-reduction works [71, 103, 167, 164]. Herein, 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.

4.4.1 *Distance preservation error*

As outlined in Sec. 4.3, our projection errors are implicitly caused by wrong realizations of distances in the projected dataset – that is, distances between points in 2D which are not proportional to their nD counterparts. To measure such errors, we compute the projection error of point i *vs* a point $j \neq i$ as

$$e_{ij} = \frac{d^m(\mathbf{q}_i, \mathbf{q}_j)}{\max_{i,j} d^m(\mathbf{q}_i, \mathbf{q}_j)} - \frac{d^n(\mathbf{p}_i, \mathbf{p}_j)}{\max_{i,j} d^n(\mathbf{p}_i, \mathbf{p}_j)}. \tag{4.2}$$

By definition, $e_{ij} \in [-1, 1]$. In detail, values $e_{ij} < 0$ show projected points which are too close with respect to their nD counterparts – *i.e.*, false neighbors. Values $e_{ij} > 0$ show projected points which are too far apart with respect to their nD counterparts – thus, missing neighbors. Zero values indicate points which are projected optimally with respect to preserving their nD distances.

4.4.2 *Visualizing aggregated error*

Our first step to understanding projection errors implied by Eqn. 4.2 is to summarize the values e_{ij} by reducing them to a single value per projected points. For this, we compute for each point i a so-called aggregate error

$$e_i^{aggr} = \sum_{j \neq i} |e_{ij}|. \tag{4.3}$$

Intuitively, e_i^{aggr} captures the wrong placement, or projection error, of point i with respect to *all* other points $j \neq i$. Interpreting this metric is simple: Low e^{aggr} values correspond to points whose projections can be reliably compared with most other projected points in terms of assessing similarity as captured by their inter-point distance. Such well-placed points are good candidates for representative points used by multilevel projection methods such as [69, 171, 34, 163]. In contrast, points having large values of e^{aggr} are badly placed with respect to most other points. Such points are good candidates for being replaced, *e.g.*, by local projection-optimization techniques [168, 167].

A first, simple, way to visualize e^{aggr} is to color-code it onto the 2D projection, *e.g.* using a blue-yellow-red diverging colormap [26] (see Fig. 4.1 (a)). This image allows locating hot-spots having either very low or very high projection errors. However, finding *zones* of points having specific error ranges can be hard using a color-coded scatterplot, as points are either too small to be easily visible, or, if drawn larger, they may cause clutter by overlaps. More specifically, we are not interested in viewing this information at point level, but to (a) find compact zones in the projection having similar e^{aggr} values, and (b) find outliers, in terms of e^{aggr} values (if any). To this end, we propose a space-filling visualization, as follows. Let $DT(\mathbf{x} \in \mathbb{R}^2) = \min_{\mathbf{q} \in D^m} \|\mathbf{q} - \mathbf{x}\|$ be the distance transform

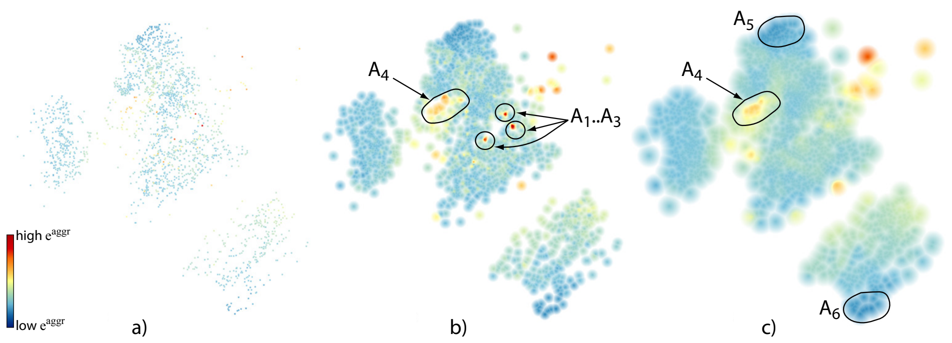


Figure 4.1: Aggregate error view showing three detail levels: (a) $\alpha = 1, \beta = 1$. (b) $\alpha = 5, \beta = 5$. (c) $\alpha = 20, \beta = 20$ pixels (see Sec. 4.4.2).

of the 2D point cloud D^m. Formally speaking, $DT(\mathbf{x}$ gives, for any pixel $\mathbf{x}$, its distance to the closest point in the projection D^m. We then extrapolate e^{aggr} at every screen pixel $\mathbf{x}$ from its values recorded at the projected points $\mathbf{q}$ as

$$e^{aggr}(\mathbf{x}) = \frac{\sum_{\mathbf{q}\in N_\epsilon(\mathbf{x})} \exp\left(-\frac{\|\mathbf{x}-\mathbf{q}\|^2}{\epsilon^2}\right) e^{aggr}_{\mathbf{q}}}{\sum_{\mathbf{q}\in N_\epsilon(\mathbf{x})} \exp\left(-\frac{\|\mathbf{x}-\mathbf{q}\|^2}{\epsilon^2}\right)} \tag{4.4}$$

with

$$\epsilon = DT(\mathbf{x}) + \alpha. \tag{4.5}$$

In the above, $N_\epsilon(\mathbf{x})$ denotes the neighborhood containing all point projections in D^m found within a radius ϵ from the pixel $\mathbf{x}$. We visualize $e^{aggr}(\mathbf{x})$ as an RGBA texture, where the RGB color components encode $e^{aggr}(\mathbf{x})$ mapped via a user-chosen color map, and the transparency A is defined as

$$A^{aggr}(\mathbf{x}) = \begin{cases} 1 - \frac{DT(\mathbf{x})}{\alpha}, & \text{if } DT(\mathbf{x}) < \beta \\ 0, & \text{otherwise} \end{cases} \tag{4.6}$$

This design allows us to obtain a continuous range of simplified visualizations of the aggregated error by changing the parameters α and β, as follows: For $\alpha = 1, \beta = 1$, we get the basic colored scatterplot shown in Fig. 4.1 (a). For $\alpha = 1, \beta > 1$, the space between the projected points is filled, up to a distance β, by the value e_i^{aggr} of the closest projected point i to the current pixel. For $\alpha = 1, \beta = \infty$, we obtain essentially a Voronoi diagram of the projected points, whose cells are colored by their e^{aggr} values. This simply extrapolates the e^{aggr} data values to larger spatial extents than individual pixels, thus making them easier to see, especially in case of outlier error values. Note that the obtained visualizations are (conceptually) similar to those we would obtain by drawing a scatterplot with points drawn as disks with radii equal to β, but avoids having the issues created by overlapping points in case β is larger than inter-point distances. For $\alpha > 1, \beta > 1$, we obtain a result similar to Shepard interpolation of the values e_i^{aggr}, where the interpolation kernel size ϵ 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 which are less interesting. Larger α values create a smoother signal where coarse-scale error patterns are more easily visible.

Figs. 4.1 (b,c) illustrates the aggregate error visualization for the Segmentation dataset for two values of the parameters α and β. For this dataset, $e_{ij} \in [-0.67, 0.35]$. This error range tells us that we have poorly projected points, but does not tell *where* these are. Using low values for both α and β, Fig. 4.1 (b) shows us that e^{aggr} is relatively evenly distributed over the whole projection. We also see three small red spots $A_1..A_3$. These indicate high-error outlier zones, which contain points badly placed with respect to *most* other points. We also notice a relatively high error large zone A_4. Larger values of both α and β simplify this visualization by removing small-scale spatial variations (Fig. 4.1 (c)). In contrast, larger β values fill in the whitespace between points. Larger α values smooth out outlier regions whose size is smaller than α, like the three small outlier areas $A_1..A_3$ discussed earlier. In contrast, A_4 remains visible, as it is larger than α. We now also notice, better than in Fig. 4.1 (b), two dark-blue zones (A_5, A_6) in the projection. These have a significantly lower error than the rest of the projection.

It is interesting to compare our visualizations with the dense *pps* maps of Schreck *et al.* [197] (see Sec. 4.2). While the aim of the two techniques is related (showing local projection errors), several differences exist: (1) Our e_i^{aggr} is a *global* metric, that tells how point i is placed with respect to *all* other points; in contrast, the *pps* metric characterizes *local* neighborhoods. (2) Our technique, for $\alpha = 1, \beta = \infty$, yields the same Voronoi diagram as Schreck *et al.*, which is also found in [10, 127]. However, we show different error values: Our e^{aggr} shows the sum of distance compression and stretching metrics introduced in [10, 127], whereas [10, 127] show these two quantities separately. (3) Both Schreck *et al.* and our method use smoothing to remove small-scale details. However, Schreck *et al.* uses a constant-radius smoothing kernel which blurs the resulting visualization equally strong everywhere. In contrast, we use, as explained, a variable-radius kernel controlled by the local point density, which is more suitable for preserving detail in non-uniform-density scatterplots.

4.4.3 *Visualizing false neighbors*

However useful to assess the error distribution and find badly *vs* well-projected point groups, the aggregate error view does not tell us how this error occurs, *i.e.*, whether it is caused by false neighbors, missing neighbors, or a mix of both. To refine insights, we next consider false neighbors (case **A**, Sec. 4.3). We show these by building a Delaunay triangulation of the projected point cloud. By construction, this gives us the closest neighbors, in all directions, of each projected point, *i.e.*, the most important potential false-neighbors 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 $e_k^{false} = |\min(e_{ij}, 0)|$. This essentially maps the false-neighbor errors between the central point of a triangle fan and the points on the periphery of the fan. Next, we interpolate e^{false} over all pixels $\mathbf{x}$ of each triangle T by using

$$e^{false}(\mathbf{x} \in T) = \frac{\sum_{1 \leqslant k \leqslant 3} \frac{1}{d(\mathbf{x}, E_k)\ \|E_k\|} e_k^{false}}{\sum_{1 \leqslant k \leqslant 3} \frac{1}{d(\mathbf{x}, E_k)\ \|E_k\|}} \tag{4.7}$$

where $d(\mathbf{x}, E)$ is the distance from $\mathbf{x}$ to the edge E and $\|E\|$ is the length of the edge. We next render an image-based view for e^{false} as an RGBA texture, similarly to the approach used for showing the aggregated error. However, we now use a heated body colormap [26], where light hues map low e^{false} values and dark hues map high e^{false} values respectively. This way, the user's attention of attracted to high e^{false} values, which are arguably more interesting. The transparency A is given by

$$A^{false}(\mathbf{x}) = A^{aggr}(\mathbf{x}) \left(1 - \frac{1}{2}\left(\min\left(\frac{DT_T(\mathbf{x})}{DT_C(\mathbf{x})}, 1\right) + \max\left(1 - \frac{DT_C(\mathbf{x})}{DT_T(\mathbf{x})}, 0\right)\right)\right) \tag{4.8}$$

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 T's barycenter, and A^{aggr} is given by Eqn. 4.6. Note that Eqn. 4.8 was originally proposed in a different context to smoothly interpolate between two 2D nested contours [187], and further adapted to interpolate between two nested shapes [216]. We refer to the above two papers for implementation details. The combined effect of Eqns. 4.7 and 4.8 is to slightly thicken, or smooth out, the Delaunay triangulation drawing. Similar to the effect of β in Eqn. 4.6, this makes the Delaunay visualization easier to perceive in terms of edges connecting close points. Note that this interpolation does *not* change the actual values e_k^{false} rendered on the triangulation edges. Additionally, the distance-dependent transparency ensures that data is shown only close to the projection points.

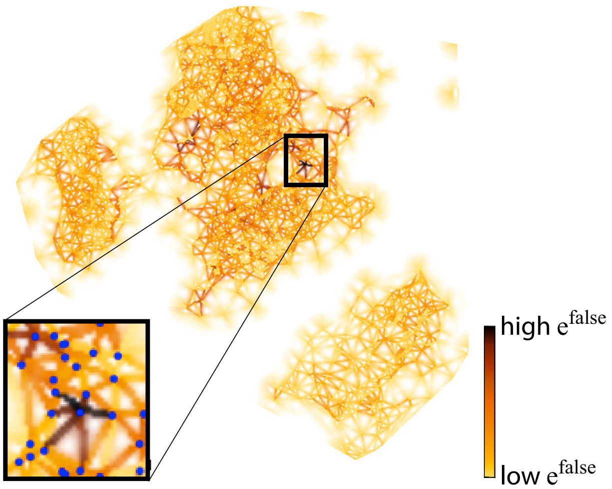


Figure 4.2: False neighbors view (see Sec. 4.4.3).

Fig. 4.2 illustrates the false neighbors view for the Segmentation dataset. We notice several aspects: First, as explained, we see a 'blurred' rendering of the Delaunay triangulation of the 2D projections colored by e^{false}, which depicts for each point which are its immediate closest neighbors. Light-colored edges show true neighbors, while dark edges show false neighbors. Since edges are individually visible, 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 easier to follow than a Delaunay triangulation drawn with pixel-thin edges, as our edges appear slightly thicker, thus more visible. However, as for the interpolation proposed for the aggregate error, no clutter is created (in terms of overlapping thick Delaunay edges).

Figure 4.2 outlines two additional error-related insights. First, we see an overall trend from light to dark colors as we go further from the projection's border towards its center. This is in line with the known fact that DR methods tend to have a lower error (in terms of false neighbors) on their borders, as there is more space to place points there. In contrasts, for points deeply embedded in the projection, there is less freedom to arrange them in terms of their neighbors, so these get a higher chance of having false neighbors (or being false neighbors of other points, the false-neighbor relation being symmetric). Intuitively, we can think of this phenomenon as a 'pressure' which builds up within the projected point-cloud from its border inwards. Section 4.5 shows several additional examples of this situation. Secondly, we see a few small-scale dark outliers in Figure 4.2. Zooming in, we discover that these are points connected by dark edges to most of their closest neighbors in a star-like pattern. Clearly some points are wrongly placed here. These can be either the star 'center' or the tips of its branches. We also see that these tips have only one dark edge – the one going to the star center. Hence, they are too closely positioned to the star center *only*, and not to their other neighbors. In contrast, the star center is wrongly placed with respect to all its surrounding points (the tips). Hence, we conclude that too little space was offered in the projection to the center point, or in other words that the center point is a badly placed point with respect of its surrounding points (and not conversely).

Our false neighbors visualization is related to Aupetit's segment-compression view, which shows the reduction of inter-point distances due to projection [10]. The underlying metrics used to capture distance-preservation errors, *i.e.* our e_{ij} (Eqn. 4.2) and m_{ij}^{distor} ([10], Sec. 3.2) are similar, up to different normalizations. The visualizations used for these metrics are, however, different: Aupetit uses so-called 'segment Voronoi cells' (SVCs). SVCs essentially achieve piecewise-constant 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. 4.7) is C^{∞}-continuous over T. Also, our triangles are increasingly transparent far away from their edges (Eqn. 4.8). Comparing our results (*e.g.* Figs. 4.2, 4.9 (a,d,g)) with SVCs (*e.g.* Figs. 7 (d), 12 (c) in [10]), we see that SVCs show 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. A similar problem and discussion thereof is present in a different context where Voronoi cells were used to visualize scatterplot data obtained from multidimensional projections [28]. Such edges do not exist in our visualization due to our continuous interpolation.

4.4.4 *Visualizing missing neighbors*

As explained earlier, projection errors also encompass missing neighbors (case **B**, Sec. 4.3). Showing these by a space-filling method like for the aggregate error or false neighbors is, however, less easy. The missing neighbors of a projected point $\mathbf{q}$ can be *anywhere* in the projection, and are actually, by definition, far away from $\mathbf{q}$. Visualizing these, thus, implies some way to show correspondences between far-away points in a projection.

One way to avoid this complication, and still use space-filling methods, is to restrain the aim of the visualization: Given a *single* point $\mathbf{q}_i$, show which of the other points $D^m \setminus \mathbf{q}_i$ are its missing neighbors. The point $\mathbf{q}_i$ can be selected by 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 show $e^{missing}$ by the same image-based technique used for the aggregated error (Sec. 4.4.2).

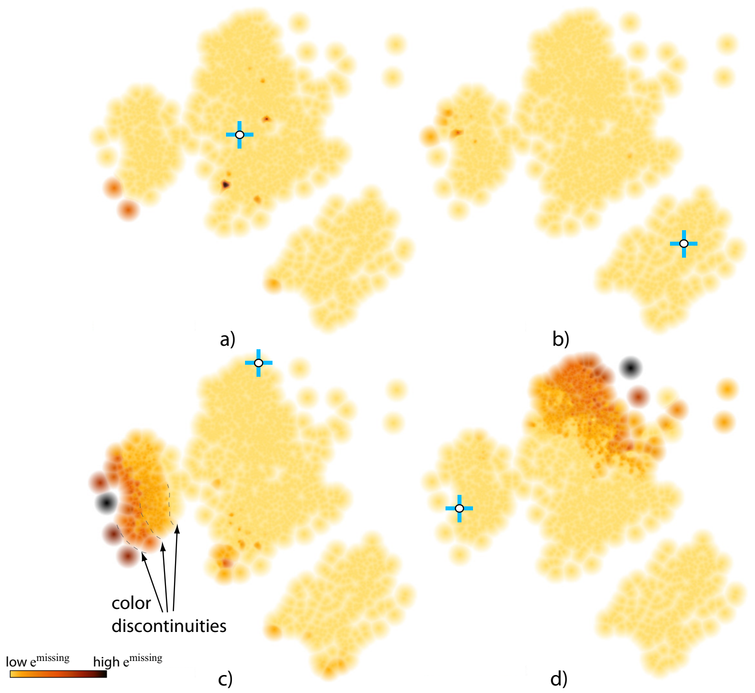


Figure 4.3: Missing neighbors view for four selected points (indicated by markers). See Sec. 4.4.4.

We use again the Segmentation dataset to illustrate the missing neighbors for a single selected point (Fig. 4.3). Here, we color map $e_i^{missing}$ via the same heat colormap as in Fig. 4.2. In Figs 4.3 (a,b), we selected two points deep inside the central, respectively the lower-right point groups in the projection. Since Figs. 4.3 (a,b) are nearly entirely light-colored, it means that these two selected points have few missing neighbors. Hence, the 2D neighbors of the selected points are truly *all* the neighbors that these points have in nD. In Figs. 4.3 (c,d), we next select two points close to the upper border of the large central group and the left border of the left group in the projection, respectively. In contrast to Figs. 4.3 (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, and they are actually more similar than the projection suggests. This is a known (but, to our knowledge, never visualized as such) challenge of many DR methods when embedding high-dimensional manifolds in 2D: points close to the embedding's *border* are too far away from other points in the projection. Intuitively, this can be seen as the projection method (LAMP) trying to 'spread' the projected points more than strictly necessary, likely in order to better reduce false neighbor issues (discussed earlier). Since, as explained earlier too, there is more room to arrange points at the border of a projection, missing neighbors likely appear in these border areas too.

Another interesting insight is that Figs. 4.3 (c,d) do not show a *smooth* color gradient as we go further from the selected point: Especially in Fig. 4.3 (c), we see that colors appear grouped in several 'bands', separated by discontinuities. In other words, the projection method (LAMP) increases the error non-uniformly with the distance.

Our missing neighbors visualization is related to the proximity view of Aupetit [10], as follows: Both views allow selecting a point i and next compute a scalar error metric (related to the selected point) and next draw it on all points $j \neq i$. For Aupetit, this error metric is the distance $m_j^{prox} = d^n(\mathbf{p}_i - \mathbf{p}_j)$ (normalized by its maximum). For us, this is the error $e_j^{missing}$. Both m^{prox} and $e^{missing}$ tend to be small at points j close in 2D to the selected point i, and increase farther off from point i. Yet, the two metrics are different and serve different purposes. Visualizing m^{prox} helps locating points found within some 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 showing the absolute d^n distance. Hence, our visualization helps locating projection errors rather than assessing proximity.

4.4.5 *Missing neighbors finder*

The visualization in Sec. 4.4.4 cannot show missing neighbors for an entire dataset, but only for a single selected point. This hampers a quick and easy understanding missing neighbors, as, in theory to find all these, one would need to iteratively select all points in a projection and retrieve (and remember) all their missing neighbors.

To address this, we proceed as follows. Consider all values $e_{ij} > 0$. These give all point-pairs which are projected too far away, *i.e.*, which are mutual missing neighbors. We sort the set containing these values decreasingly, and select the top ϕ percent of them, where ϕ is a user-provided parameter. The selected values give the point-pair-set $MN = \{(\mathbf{q}_i, \mathbf{q}_j)\}$ which are the worst mutual missing neighbors. We next construct a graph $G = (V, E)$ whose nodes V are the projected points $\mathbf{q} \in MN$, and edges E indicate the pairs $(\mathbf{q}_i, \mathbf{q}_j)$, with e_{ij} mapped to edge weights. Next, we draw a simplified version of G using the KDEEB edge bundling technique [93], which is a robust, easy to use, real-time bundling technique for graphs with tens of thousands of edges. Finally, we color the bundled edges based on their weights e_{ij} via a grayscale colormap (with white mapping low and black mapping high weights respectively), and draw them sorted back-to-front on e_{ij} and with an opacity proportional to e_{ij}. The most important edges thus appear atop and opaque, and the least important ones are at the bottom and transparent. This visualization is called further the *missing neighbors finder*.

We can use the missing neighbors finder in two different modes, as follows.

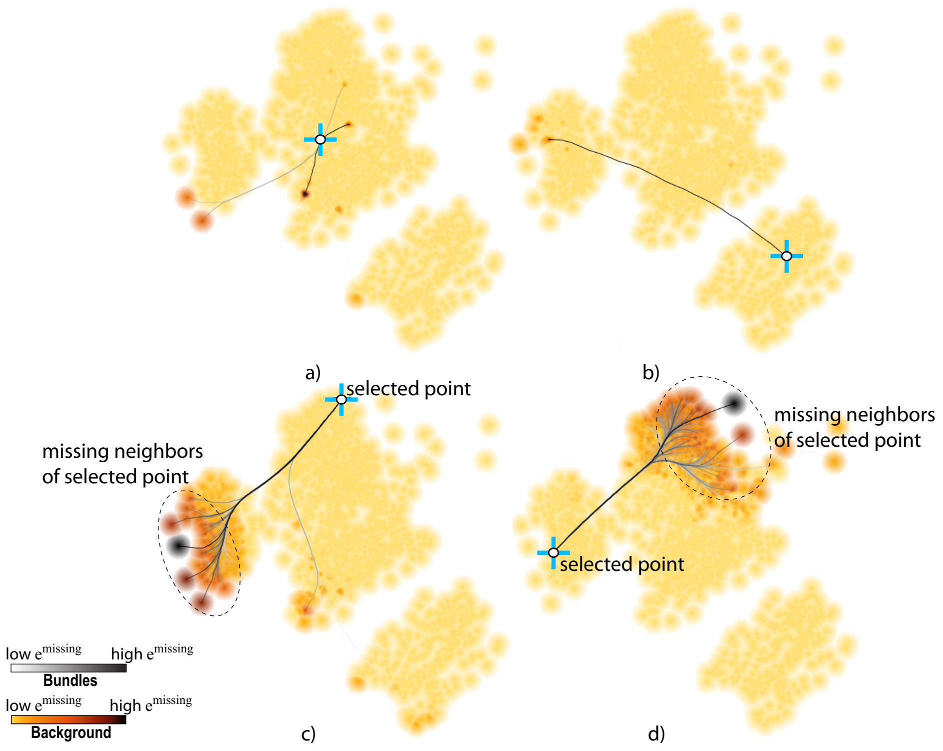


Figure 4.4: Missing neighbors finder view for four selected points. Selections are indicated by markers (see Sec. 4.4.5).

First, we can restrict the construction of the graph G to include only edges that have, at one end, a selected point $\mathbf{q}_i$. This point is selected precisely as described in Sec. 4.4.4. As such, the bundled version of G will contain only edges that start, or end, at the location $\mathbf{q}_i$. Fig. 4.4 shows this for the same four selected points discussed earlier in Fig. 4.3. For comparison, the background in Fig. 4.4 shows $e^{missing}$ (Sec. 4.4.4). 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. Compared to the earlier technique for showing missing neighbors (of a selected point) by means of color mapping (Fig. 4.3), bundles have the added value of focusing the attention of the user to the *most important* missing neighbors (shown as dark edges); highlighting explicitly the fact that these are missing neighbors of the selected point; and allowing one to better assess the 2D distances between the selected point and its missing neighbors, and the spread of the latter over the projection.

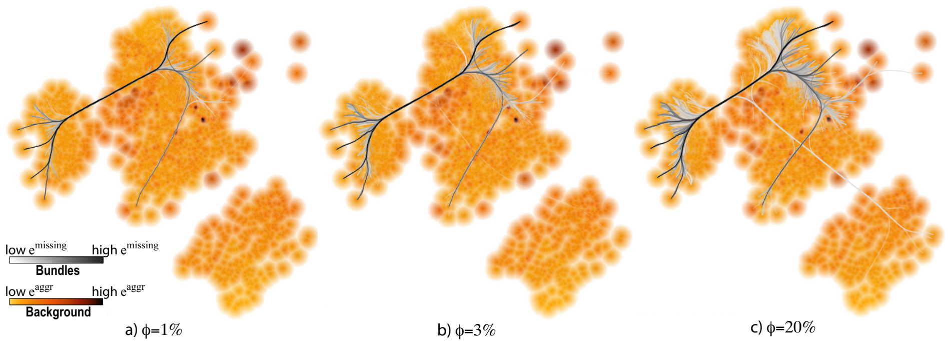


Figure 4.5: Missing neighbors finder view, all point pairs, for different ϕ values (see Sec. 4.4.5).

The second mode of the missing neighbors finder extends to showing many-to-many missing-neighbor relations between all projected points. For this, we use the aforementioned procedure of constructing G without restricting its edges to start or end at the same single selected point. Fig. 4.5 shows this result for three values of ϕ for the Segmentation dataset. The background shows now the aggregated error (e^{aggr}, Sec. 4.4.2). We now color bundles from black for largest error e_{ij} to white for largest error above the user-provided parameter ϕ. Image (a) shows the $\phi = 1\%$ worst missing-neighbor point-pairs. These link the top-right area of the central group with the left border of the left group. Adding more missing neighbor pairs to the view, by increasing ϕ, (image (b), $\phi = 3\%$) strengthens this finding. Adding even more missing neighbor-pairs (image (c), $\phi = 20\%$) reveals additional such pairs between the two zones 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 earlier hypothesis that such point pairs are challenging for the LAMP projection, which we already noticed when using the missing neighbors view (Sec. 4.4.4). However, as compared to that view, the bundled view shows all such point pairs in a single go, without requiring user interaction.

Several technical observations are due here. First, the usage of edge bundling is motivated by the aim to decrease clutter that would be formed when drawing a straight-line node-link view of the graph G – which is typically unavoidable, since missing neighbor-pairs are located, by definition, far away from each other in the projection. Secondly, edge bundling creates a simplified view that highlights the coarse-scale missing-neighbor connectivity patterns occuring over *groups* of closely-placed points in the projection. Thirdly, typical settings of the parameter ϕ range between 1% and 10% of the total amount of point-pairs in a projection. Showing more point-pairs can be technically done with no problems, since the bundling algorithm used is highly scalable to graphs of hundreds of thousands of edges [93]. However, the added-value of doing this is limited, since only a small fraction of the point-pairs in a typical projection are badly placed with respect to each other (at least, this is true for projections generated by high-quality algorithms, such as LAMP). Fourthly, an alternative way to show the most important missing neighbor-pairs is to threshold the value-set $e_{ij} > 0$ by an absolute error value rather than a percentage. This easily allows us to *e.g.* compare different projections in terms of the amount, and distribution, of their missing neighbors.

4.4.6 *Group analysis*

The false and missing neighbors issues for individual points become, at group level, the problems of false and missing group members respectively (Sec. 4.3). To address these issues at group level, we introdude two subsequent visualizations, as follows.

First, we must clarify the concept of a point group: A group $\Gamma \subset D^m$ is a set of projected points which form a *visually* well-separated entity. That is, a group is visually separated, in a projection, by significantly large amounts of white space from other groups. Assuming that the underlying projection technique reasonably preserves distances and/or neighborhoods, visual groups indicate points that share some similarities, and are in the same time different from other visual groups.

To illustrate this, consider the LAMP projection of our Segmentation dataset. We see here three such groups (Figs. 4.1-4.5).

To further reason about such groups, we need a way to let users select them. For this, we provide several mechanisms: direct interactive selection, mean-shift clustering [44], and upper thresholding of the point density [66]. The first method is basically the most flexible but also the most time-consuming, as it requires user interaction to delineate points that the user perceives as belonging together. The second method is a (well known) clustering method that employs the variations in point density to segregate areas of high-density points separated by areas of low-density points. The third method is basically a cheaper, but less accurate, automatic clustering version of the second method. Other user-controlled methods can be used if desired, *e.g.*, K-means or hierarchical agglomerative clustering *e.g.* [102, 101]. The actual group selection mechanism is further of no importance to our visualization method. The only requirement here is that we use a method to separate the projected points into distinct groups. A separate comment is due here: One could argue that clustering points into groups using the projection is sub-optimal with respect of doing the same using the original high-dimensional points. From a pure data analysis viewpoint, aiming at finding groups of related observations, this is true. However, our entire scope (in this chapter) is not to partition the high-dimensional observations into similar sets, but to explain the *projection* of these observations. As such, when talking about groups, we consider those formed in the projection, and not those implied by the high-dimensional data. This is also the main reason why, when talking about groups, we refer to *visual groups*.

To better reason about visual groups, we next render each obtained group $\Gamma = \{\mathbf{q}_i\}$ by the shaded cushion technique in [216], 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 δ in Γ, following [44, 93]. Next, we compute a threshold-set Γ_δ of ρ at level δ, and its distance transform DT_{Γ_δ}. Finally, we render an RGBA texture over Γ_δ, where we set the color a fixed hue (light blue in our case) and the transparency A to $\sqrt{DT_{\Gamma_\delta}}$. The global effect is to create a dark-to-bright border cushion-like profile over a group, that has constant thickness, and thus better emphasizes the containment of points in a group than when drawing a simple isocontour of ρ at level δ.

With groups being now modeled as a data structure (Γ_δ) and also visually rendered, we modify the missing neighbors and finder techniques (Secs. 4.4.4, 4.4.5) to show missing group members. In detail, we compute a value

$$e_\Gamma^{missing}(\mathbf{q}_i) = \begin{cases} \min_{\mathbf{q}_j \in \Gamma}(e_{ij}) & \text{if } \mathbf{q}_i \notin \Gamma \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

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

Fig. 4.6 (a,b) show two missing group members views for our Segmentation dataset. The shaded cushions show the three groups identified by the method presented earlier (using here the thresholding of the density ρ; using mean-shift clustering gives similar results). We see that several points fall outside of these three groups. This is expected, and indicates points which are too far away from other densely-packed points to be easily assigned to a visual group. In image (a), we select the bottom group Γ_{bottom}. The underlying color map shows now $e_{\Gamma_{bottom}}^{missing}$, (Eqn. 4.9). All points appear here light yellow. This means that, no points are projected too far from Γ_{bottom}, *i.e.*, Γ_{bottom} has no missing members. In image (b), we do the same for the left group Γ_{left}. The image now appears overall light yellow, except for a small dark-red spot in the upper-right corner of Γ_{center}. Here are a few points which are placed too far from any point in Γ_{left}. These are highly likely to be missing members of Γ_{left}. To obtain more insight, we now use the bundle view introduced in Sec. 4.4.5, 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 (so that we do not miss any potential group members). Image (c) shows the bundle view for Γ_{bottom}. We see only a few bundled edges, ending at a small subset of

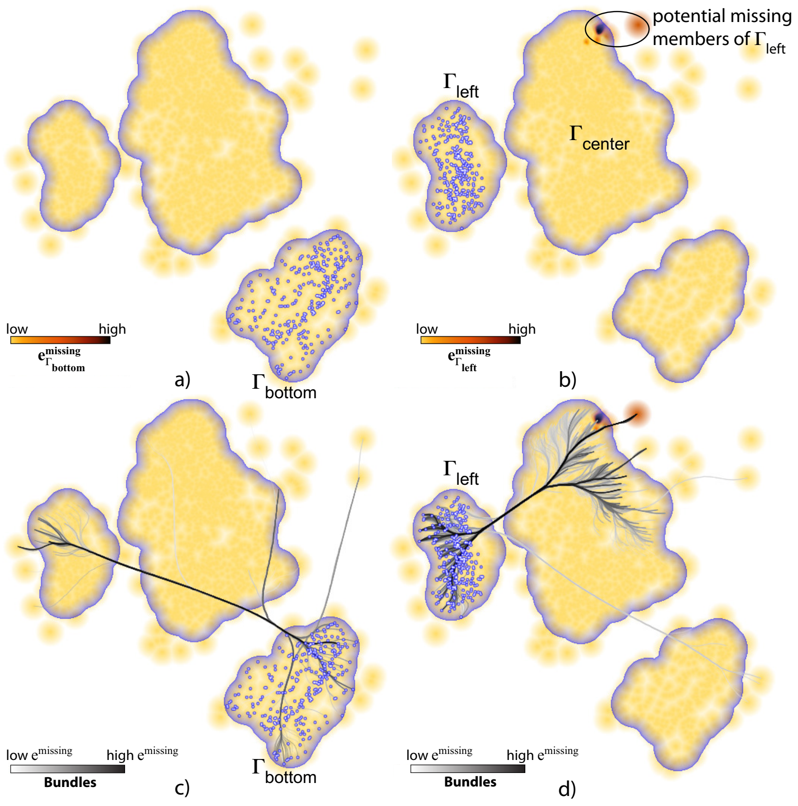


Figure 4.6: Missing members for two groups Γ_{bottom} and Γ_{left}. Points in these groups are marked by blue dots (see Sec. 4.4.6). Top images show the error metric $e_{\Gamma}^{\mathrm{missing}}$. Bottom images add edge bundles to indicate the most important missing point-pairs.

the points in Γ_{bottom}. This strengthens our hypothesis that there are no points outside Γ_{bottom} which should have been placed closer to *any* point in Γ_{bottom} – hence, Γ_{bottom} has no missing members. Image (d) shows the bundled view for Γ_{left}. The bundle structure tells us that the top-right part of Γ_{center} contains many missing neighbors of Γ_{left}. In particular, we see dark bundle edges that connect to dark-red points. This is a strong sign that these points can indeed be missing members of Γ_{left}. To refine this insight, we let the user interactively query the discovered points' details (attribute values) and, depending on these, finally decide if these points are missing group members or not.

4.4.7 *Comparing projections*

When constructing a projection, users have the choice between using different DR methods, or the same method with different parameters. The question is:

Which of the resulting projections is better, in terms of quality? The refinement of this question is: How to compare different projections?

To answer such questions, we propose a *projection comparison* visualization. The view gets as input two projections D_1^m and D_2^m of the same input dataset D^n, both based on the same control points to ensure that the projections match, even in their orientation, in case the respective projections use control points. In other case, simple rigid alignment of the projections can be done, to bring them as close as possible with respect to rotations, scalings, and translations.

For each point-pair $(\mathbf{q}_i^1 \in D_1^m, \mathbf{q}_i^2 \in D_2^m)$, we next compute a displacement

$$e_i^{disp} = \frac{\|\mathbf{q}_i^1 - \mathbf{q}_i^2\|}{\max_i \|\mathbf{q}_i^1 - \mathbf{q}_i^2\|}. \tag{4.10}$$

We next build a graph G whose vertices are points in $D_1^m \cup D_2^m$. Edges connect point-pairs $(\mathbf{q}_i^1 \in D_1^m, \mathbf{q}_i^2 \in D_2^m)$, weighted by the values e^{disp}, *i.e.*, highlight points representing the same observations which are placed, in 2D, far away from each other in the two projections. We visualize G using edge bundling, as for the missing neighbors finder (Sec. 4.4.5).

Fig. 4.7 (a) the use of this technique to compare the Segmentation dataset projected via LAMP (red points, D_1^m) and LSP (green points, D_2^m). The two projections are, at first sight, quite similar, since red and green points are in general placed close to each other. Yet, this image does not tell if the two projections create the same *groups* of points, since we do not know how red points correspond to the green ones. In other words, we do not know if closely-placed red and green points correspond to the same observations. Fig. 4.7 (b) shows the projection comparison view, where corresponding points are linked by edge bundles. We here 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 zones, apart from the already-discussed dark bundle. So, 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 (when creating A_{LAMP}), then the choice of LSP to move the point-group A all the way up in the image (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.

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

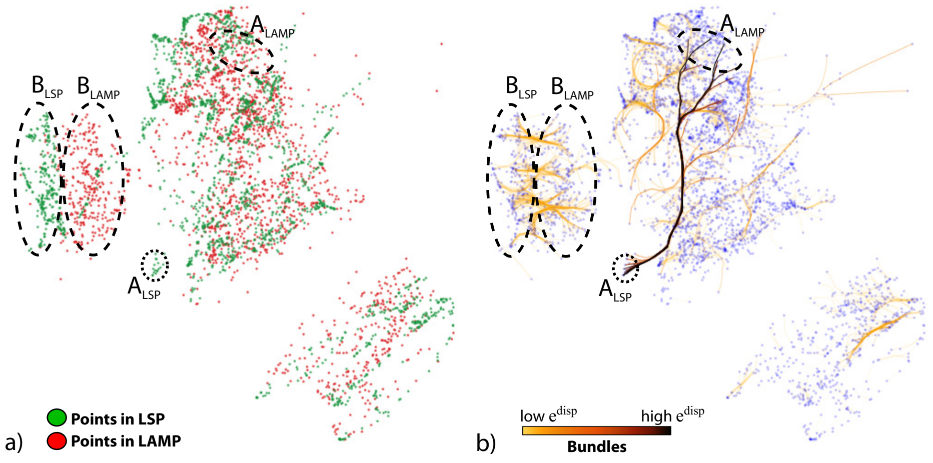


Figure 4.7: Comparing projections. (a) LAMP (blue) and LSP (red) points. (b) Bundles show corresponding point groups in the two projections (see Sec. 4.4.7).

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.

4.4.8 *Proposed workflow*

So far, we have presented several views for analyzing projection errors. Each such view has its specific focus (on a separate class of errors, related to separate analysis goals). As such, the question arises of how to combine all these views into a coherent *usage scenario* for an analysis task? Below we propose such a usage scenario.

STEP 1 Start with the *Aggregated Error* view. This way, one gets a global idea of the error at all points, without a distinction between false or missing neighbors. Next, find if (a) there are any regions with large errors or (b) the overall error is low. If we see (b), the projection is quite good (low error) and thus nothing else needs to be improved prior to using the projection for analysis purposes. In case (a), continue with steps 2, 3, and 4.

STEP 2 Using the *Missing Neighbors Finder*, one net finds the most important missing neighbor point-pairs. If this view shows bundles having high error values (*i.e.*, dark-colored), there are important missing neighbors in the projection. These point-groups connected by such bundles must be further analyzed with the *Group Analysis Views*. If no such groups exist, *i.e.* the bundles are (light) gray, this tells that the projection is good and, although there are missing neighbors, they are in a minority, and thus do not affect the projection interpretation.

STEP 3 Problematic points or groups found in steps 1 and 2 should now be analyzed in more detail using the *False Neighbors* and *Missing Neighbors* views. For groups found in step 1, we should find out exactly what kind of error occurs: Are these groups (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 found from the beginning: They have many missing neighbors. Thus, the question to be answered here is: Which points are the problematic ones inside the detected groups, or where exactly do the relations (bundled edges) with the highest errors start and end from? By using the false-neighbors and missing-neighbors views, the user can find exactly which are the more problematic points (or groups), and what kind of errors these have.

STEP 4 Knowing now where exactly errors occur, we consider the next questions: (1) Are such errors really problematic for the projection interpretation? (2) Do they show unexpected results with respect to how the projection should depict its input data? (3) Are these problematic points important for the use-case at hand? If questions (1-3) are all answered by 'no', we have a good projection for our data and use-case, so our error investigation ends. If any question (1-3) answers yes, then the user must improve the projection of problematic points: If the user is a projection-designer testing the accuracy of a new algorithm, (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, or is not interested and/or able to understand and modify the respective algorithm, the way forward is to redo the error 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 should be next be compared with the old ones to find if errors have decreased or if they moved into a new projection-region where they are not as important for the task at hand. For the second task above, the *Projection Comparison View* is the tool of choice to use.

4.5 APPLICATIONS

Let us next demonstrate the way of working and added-value of the proposed error visualizations to analyze several projections and their corresponding parameter settings – in other words, to explore the space P that controls the creation of a DR projection. To this end, we first overview the datasets and projection methods used in this study (Sec. 4.5.1), the studied projection algorithms (Sec. 4.5.2), and their parameters (Sec. 4.5.3). Next, we use our views to explore the parameter settings of the involved projections (Secs. 4.5.4, 4.5.5).

4.5.1 *Description of Datasets*

We next consider the following datasets (in addition to the Segmentation dataset used for our running example):

Freefoto: is a dataset of 3462 images grouped into 9 classes [72]. For each image, we extract 130 BIC (border-interior pixel classification) features. Such features are widely used in image classification tasks [209].

Corel: is a dataset composed of 1000 photographs that cover 10 specific topics. Similarly to the Freefoto dataset, we extract for each image a vector of 150 SIFT image descriptors [131]. The underlying task here is to use these descriptors for automatic image classification purposes.

News: is a dataset containing 1771 RSS news feeds from BBC, CNN, Reuters and Associated Press, collected between June and July 2011. The 3731 dimensions of each news-feed were created by analyzing the text of the feed to remove stopwords, employ stemming, and use term-frequency-inverse-document-frequency counts. We manually classified the observations based on the perceived main topic of the news feed. This resulted in 23 classes. Given the imprecision of the manual classification and the restriction to have one class (topic) per observation, the classes are unbalanced. Moreover, we cannot guarantee that classes having different labels do not have a highly similar content.

Sourceforge: This dataset contains 24 software quality metrics computed on 6773 open-source C++ software projects from the *sourceforge.net* portal [33]. Metrics include classical objet-oriented quality indicators such as coupling, cohesion, inheritance depth, size, complexity, and comment density [123], averaged for all source code files within a project. A separate metric records the number of downloads of a project. The aim of this collection is to see whether the measured quality metrics correlate, in any way, with the objectively measured download count.

4.5.2 *Projection Techniques*

For our study, we considered several projection techniques, based on the availability of documented parameters, scalability, genericity, presence in the literature, and last but not least, open availability of a good and easy-to-use implementation.

LSP: The Least Squares Projection [163] uses a force-based scheme to first position a subset of so-called control points. The remaining 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 nD space to the 2D projection space.

PLMP: The Part-Linear Multidimensional Projection (PLMP) [164] addresses computational scalability for large datasets by first constructing a linear mapping of control points using a force-based method. Next, this linear mapping is used to place the remaining points, by a simple and fast matrix multiplication of the feature matrix with a linear mapping matrix.

LAMP: Aiming to allow more user control over the final layout, the Local Affine Multidimensional Projection (LAMP) [103] (described also in Chapter 3) provides a user-controlled redefinition of the mapping matrix over an initial mapping of control points. LAMP uses control points to build a family of orthogonal affine mappings, one for each point to project. LAMP cannot directly work with distance relations, *i.e.*, it needs to access the nD point coordinates. Yet, 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: A separate class of projection techniques employs various optimization strategies (see Chapter 2). Such techniques are, in general, quite computationally demanding. To improve speed, Pekalska *et al.* [171] 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 well-known ISOMAP technique [218] is an extension of classical Multidimensional Scaling (MDS). 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.

4.5.3 *Projection parameters*

Most projection techniques that work with control points use an iterative force-based algorithm, such as the one of Tejada *et al.* [214] or, more generally speaking, such as used when embedding (laying out) graphs [56]. The number of *iterations* of force-based placement influences the control points' positions, being thus, an important parameter. LSP control points are typically the centroids of clusters obtained from a clustering of the input dataset. The *number of control points* is, hence, a second important parameter for LSP. To position points around a given control point, LSP solves a linear system for that neighborhood. The neighborhood size (*number of neighbors*) is a third important parameter.

LAMP builds its affine mappings from a neighborhood of control points. The size of the control point-set used to build this mapping, expressed as a *percentage* of the size of the total point-count in the input dataset, is the key parameter here. The choice of control points and the choice of the initial projection of the control points are also parameterizable, like for LSP, PLMP, and Pekalska. Yet, in LAMP, these parameters are interactively controlled by the user according to application-specific preferences, and thus of a lesser interest for us in our current analysis.

ISOMAP, just as the previously mentioned techniques, also requires the definition of point neighborhoods. The main, and frequently only, exposed parameter of ISOMAP is the number of *nearest neighbors* that defines a neighborhood.

4.5.4 *Overview comparison*

We start our comparison of projections (and their parameters) top-down, with a global comparison of the considered projection techniques – LAMP, LSP, PLMP, and Pekalska.

For this, we consider the false neighbors, aggregated error, and most important $\phi = 5\%$ missing neighbors for the Segmentation dataset, as projected by the aforementioned four algorithms (Figure 4.8). To enable an easy comparison, color mapping is normalized so that the same colors show the same absolute values across different 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) spots showing the worse-projected points. Hence, all studied projections are quite similar in terms of distance preservation quality, at a global level. The false neighbors views (middle row) show a similar finding: Frontier points have few false neighbors (light colors), and the density of false neighbors increases gradually towards the projections' centers. While local variations exist between projections, they are quite small, meaning that all four studied projections are globally equally good from the viewpoint of not creating false neighbors. The missing neighbors view (bottom row) is quite different: Seeing the size and color of the shown bundles, we

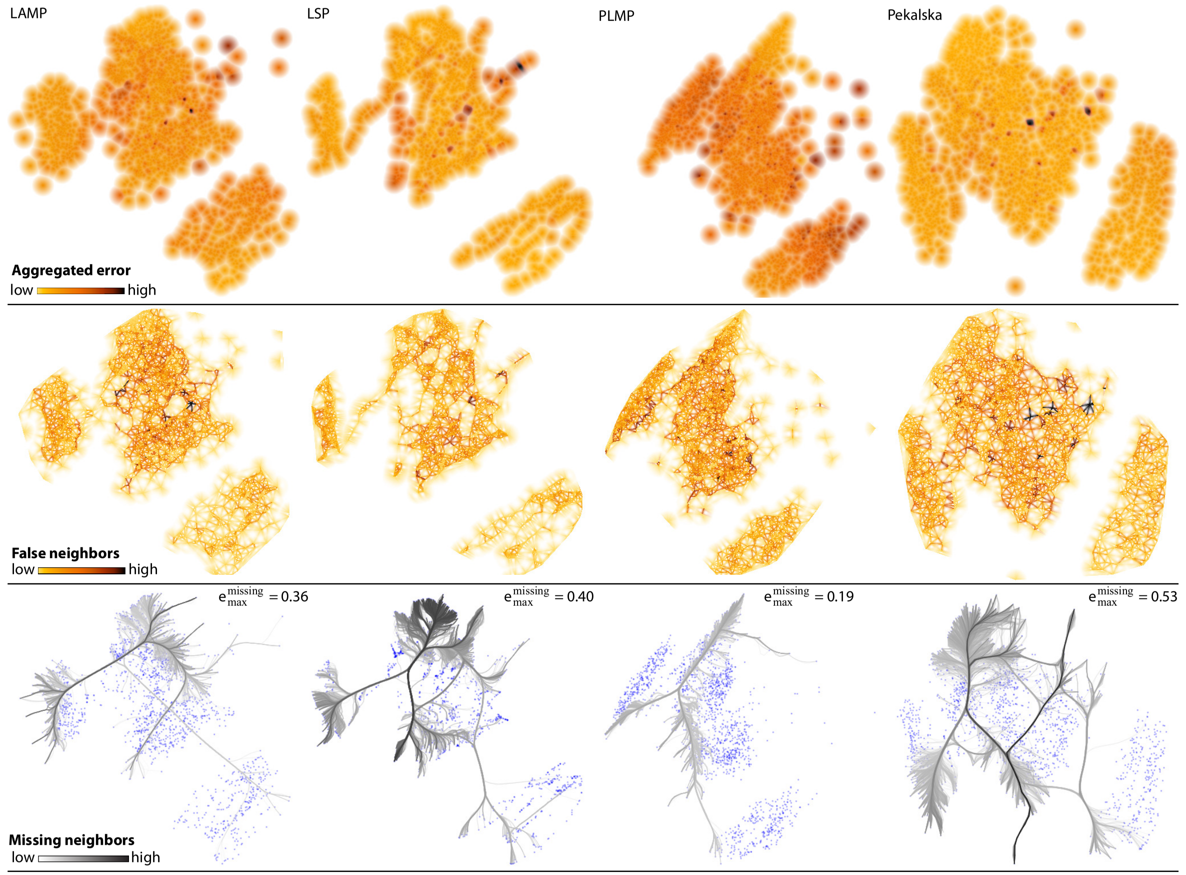


Figure 4.8: Global comparison of LAMP, LSP, PLMP, and Pekalska projections for the Segmentation dataset (see Sec. 4.5.4)
.

infer that LSP and Pekalska have much more missing neighbors than PLMP, and LAMP has the fewest missing neighbors. In all cases, we see that bundles link borders of the projected point-set. This tells that all four studied projections optimize placement of close points in the detriment of far-away points. We also see that the missing neighbors are spread differently four the four studied projections: For LAMP, there are no bundles going to the bottom-right point cluster, showing that this cluster is well separated in the LAMP projection. In contrast, LSP, PLMP, and Pekalska all have bundles going to this cluster, telling that they place these points too closely to the other projected points.

4.5.5 *Parameter analysis*

We next zoom in on the analysis of *parameters* of projection methods. To this end, we study two of the four aforementioned projection algorithms: LAMP and LSP. The restriction to these algorithms is justified by the comparatively large effort required to study all parameters of four algorithms, and separately, our interest in LAMP (part of our work, see Chapter 3), and the fact that LSP is better known in the literature than PLMP and Pekalska.

To study LAMP *vs* LSP, we next vary several of their parameters, and assess the resulting projections' quality with respect to this variation.

LAMP - Different control point percentages: Fig. 4.9 shows the results of LAMP for the Freefoto dataset with three different values for LAMP's *percentage* parameter: 10%, 30% and 50%. The error has been normalized on each column in the figure.

A first finding is that the final layout of the produced projection does not change drastically when varying the *percentage* parameter – indeed, the main change we see is a 90 degree clockwise rotation for the percentage value 30%. In the false neighbors view, we also see that, while the light brown areas are large – telling us that a moderate amount of error is globally present – the dark-colored spots are found nearer to the center. This tells that LAMP positions the most problematic points in the center. By focusing on the dark spots (having the highest false-neighbor errors) throughout the parameter variation, we can see that the largest error values remain quite similar – all views have roughly the same percentage of dark-colored areas.

Next, in the missing neighbors view, we select a point near the upper border of the projection (marked by a cross in Figs. 4.9 (b), (e) and (h)). We do this since missing neighbors occur mainly on the borders of the projection, as we have discussed in Sec. 4.4.4. The dark spot in Fig. 4.9 (h) shows the area where the largest error occurs over these three views. While in Fig. 4.9 (b) there are a few orange spots showing moderate error, in Fig. 4.9 (e) the error visibly decreases, while increasing again in Fig. 4.9 (h). This tells that using roughly 30% of neighbors is a good parameter setting to avoid missing neighbors. We confirmed this finding on several other datasets projected via LAMP (not shown

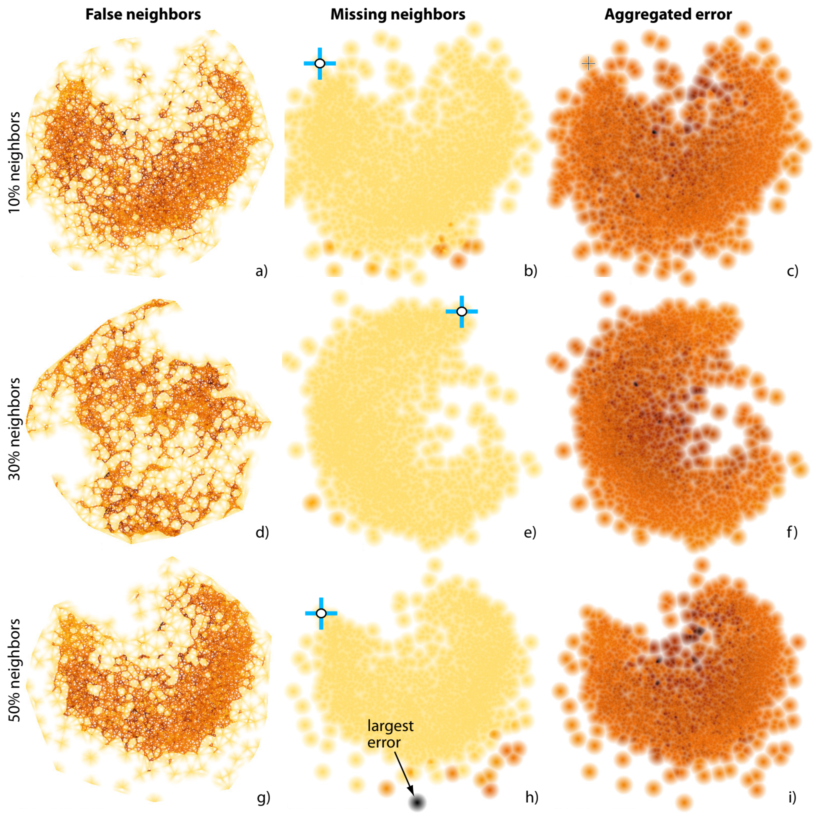


Figure 4.9: LAMP algorithm, Freefoto dataset, different neighbor *percentages* per row (see also Fig. 4.10).

here for brevity). Finally, the aggregated error view shows insights which are quite similar to the false neighbors view: More problematic points, error-wise (dark spots), are present in the projection center, while a moderate amount of error is spread evenly over the entire projection. This tells that LAMP gets most its projection errors from false neighbors rather than from missing neighbors.

LSP - Different numbers of control points: Figure 4.10 shows the Freefoto dataset projected with LSP. The varying parameter under study is now the *number of control points* of LSP. We use here the same views as in Fig. 4.9, and normalize the error in each column. The false neighbors views show a spatial mix of light-yellow and orange-brown colored areas in the projection. This contrasts with LAMP (Fig. 4.9) where the larger missing neighbor errors are consistently located far away from the projection border. When the *number of control points* increases, the large-error areas get more compact and closer to the projection center. However, we spot 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. 4.10 (b) vanish largely in images (e) and (h). This means that the severity of the missing neighbors diminishes when the number of control points increases. If we compare this with LAMP (Fig. 4.9 b,e,h), we see thus 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) are to be found in the projection center: In other words, the most problematic points are pushed inside in the projection by the 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 studied parameter values.

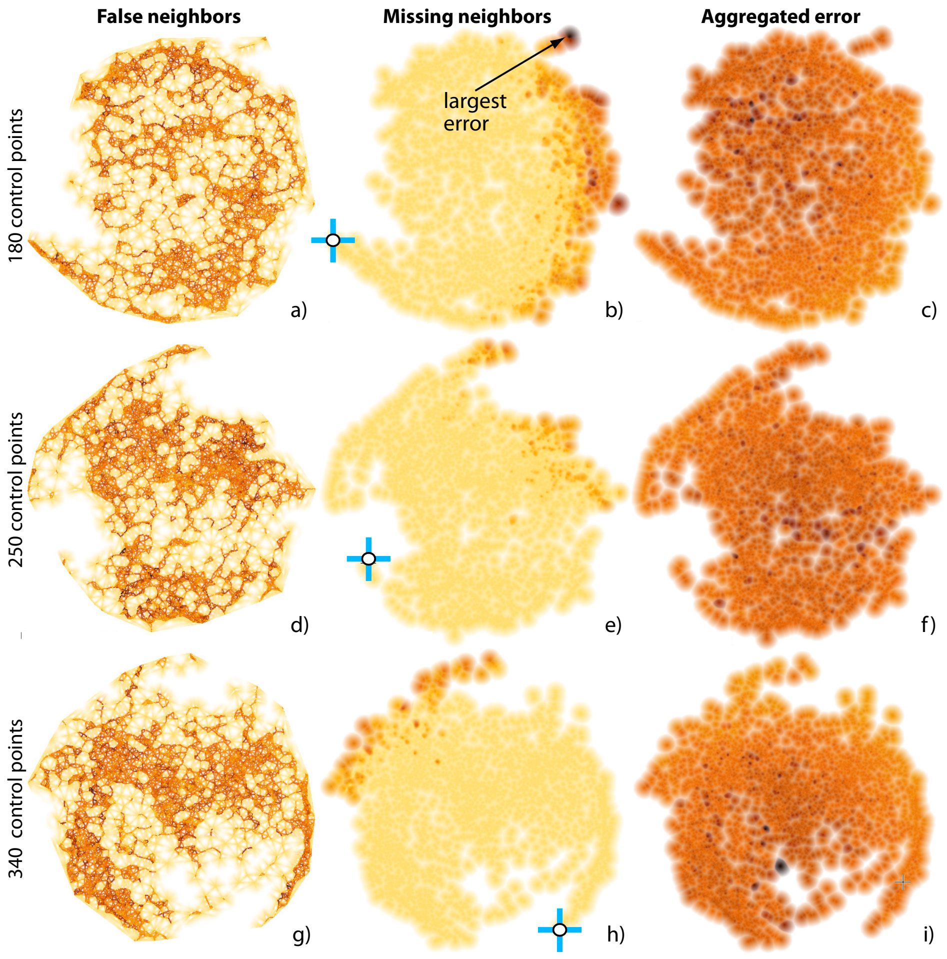


Figure 4.10: LSP technique, Freefoto dataset, different numbers of *control points* per row (compare with Fig. 4.9)

LSP - Different numbers of neighbors: Let us next study the effect of a second parameter of LSP: *number of neighbors*. For the Freefoto dataset, we fix the number of control points to 250, and next vary the number of neighbors to 10, 50 and 100, respectively. Fig. 4.11 shows the resulting projections with the missing neighbors finder view. We see here that the most significant errors are initially located between groups A, B and C, with group C showing itself to be placed too far from both A and B. Increasing our number-of-neighbors parameter has a positive impact on solving the missing neighbors problem between groups A and C, bringing them together into the group marked AC. The most important missing neighbors are now between groups AC and B. This 'concentration' of error given by the number-of-neighbors parameter increase is, upon further analysis, explainable if we consider how LSP works: Given a neighborhood N, LSP's Laplacian smoothing places all points in N close to each other in the final layout. Yet, the position of the neighborhoods N_i *themselves* is conditioned only by the control points, which are determined by the initial force-based layout. If this force-based 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, since Laplacian smoothing has only a local effect. Thus, as the neighborhood size increases, the likelihood to see fewer thick high-error bundles increases. This insight is interesting since it was not reported in the LSP papers we are aware of.

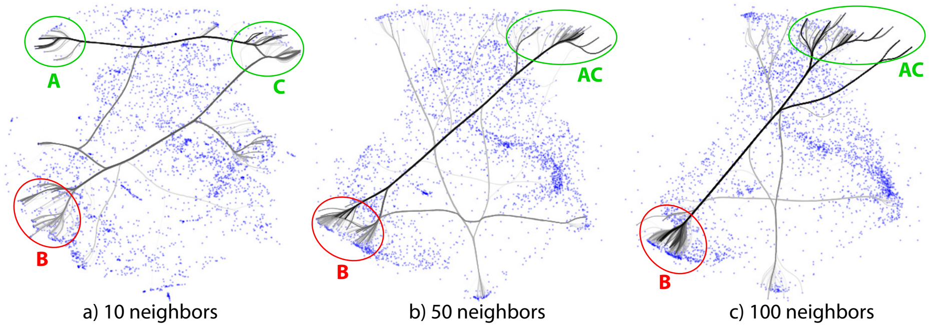


Figure 4.11: Applications – LSP technique, Freefoto dataset, different numbers of *neighbors*. Bundles show most important missing neighbors.

LAMP - Different datasets: We next analyze the LAMP projection method over three different datasets: Corel (1000 elements), Freefoto (3462 elements), and Sourceforge (6773 elements). The varying parameter is now the input *dataset* itself, rather than parameters of the projection method. By this, we want to see whether (and how) errors are influenced by the nature of the input data, *e.g.* distribution of point-distances, number of dimensions, and number of points. Figure 4.12 (top-row) shows the false neighbors views for the three considered dataets. While for the first two datasets the behavior of false neighbors is similar to earlier results, for the largest dataset (Sourceforge) we see considerably fewer false neighbors. These are found close to the intersection area of the two apparent visual groups in the image, and also on the borders of these groups. This, and the low errors (light colors) inside the groups indicate that both groups are highly cohesive. The large errors on close to the intersection areas and borders indicates elements that could be in either group, respectively very different from all other elements. Figure 4.12 (a) shows a similar insight: Most false neighbors are found at the 'star' shape's center, while the arms of the star contain more cohesive (similar) elements. This indicates that the dataset contains a number of cohesive groups equal to the number of star arms, and that elements in the center belong equally to all groups.

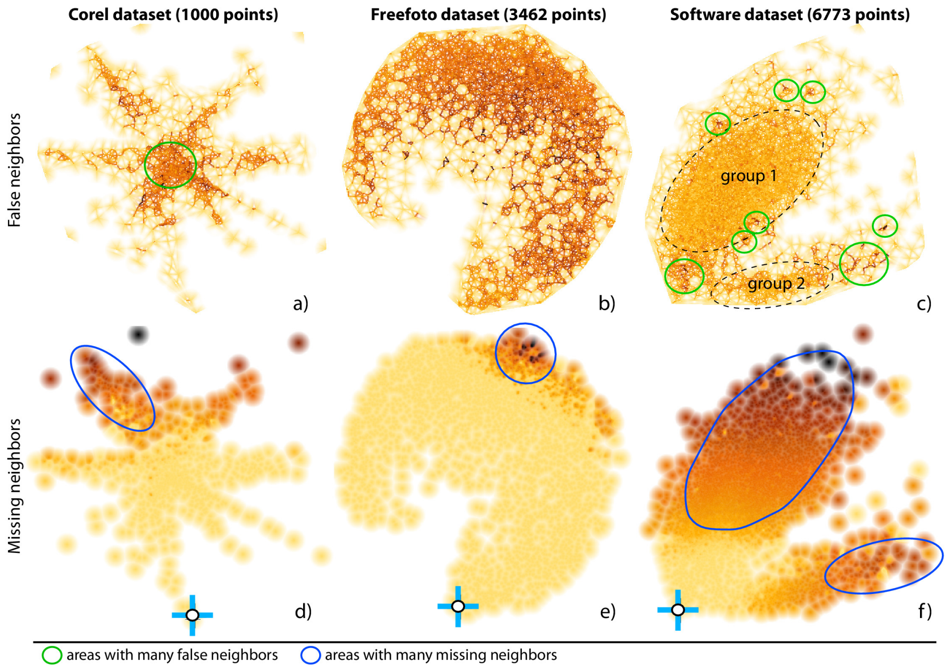


Figure 4.12: One algorithm (LAMP), different datasets. Top row: false neighbors. Bottom row: missing neighbors.

Analyzing the missing neighbors for several points selected on the border of projections, we see that errors are smaller for Figs. 4.12 (d) and (e), and quite larger for Fig. 4.12 (f). For the last image, we selected a point close to the intersection area of the perceived visual groups. Image (f) shows that this point is *equally* too far-placed from most points in *both* visual groups. The size and speed of increase of the error (as we get further from this point in the 2D projection space) suggests that the selected point stronger belongs to both visual groups than the projection tells. This strengthens our initial hypothesis that the area separating the two groups belongs equally to these groups.

ISOMAP - Different numbers of neighbors: Figure 4.13 shows the effect of changing the number of *neighbors* in ISOMAP on missing group members. Our group of interest Γ, shown first on Fig. 4.13 (a), is outlined in images (b-d) by a shaded cushion. Besides the fact that Γ moves from the left to the right of the projection, which is less interesting, images (b-d) show how its missing members behave as we change our number-of-neighbors parameter. First, Fig. 4.13 (b) shows that the most important missing neighbors are located in two other areas A_1 and A_2 on the far side of the layout. We also see many black edges, telling that points in A_1 and A_2 are indeed too far away from all points in the selected group. The quite large fan-out of the bundles show that the group misses many members, which get widely scattered over the projection. As the parameter increases, image (c) shows that the missing members spread out even more, but the severity of the errors drops (see the lighter colors of $e_{\Gamma}^{missing}$ in the background). The inner fanning of the edges in Γ is still large, telling that many group members miss neighbors. Finally, in Fig. 4.13 (d), error-related problems decrease markedly: We see thinner bundles, which mean less error; the bundle fanning in Γ is quite 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 analyzed group, the increase of the number of neighbors parameter positively influences the projection quality.

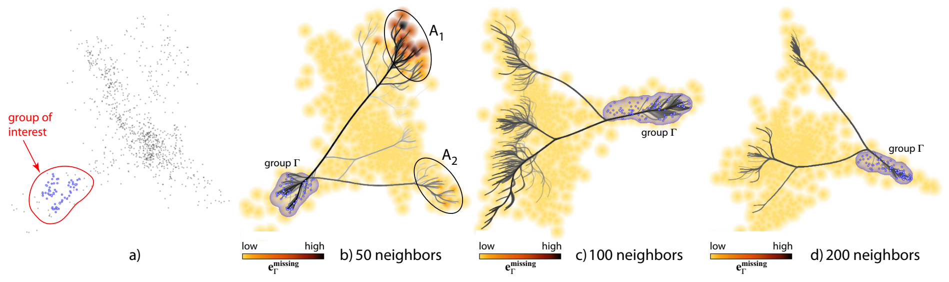


Figure 4.13: ISOMAP projection, Corel dataset, finding missing group members for different numbers of *neighbors*.

LSP - Different numbers of iterations: Our last analysis compares two different LSP projections of the same dataset (News), created using values of 50, respectively 100 for the *number of iterations* parameter of the control-point force-driven placement.

Figures 4.14 (a) and (b) show the two LSP projections for the two above-mentioned values of the number-of-iterations parameter. Both projections show several high-density groups. These are strongly related news feeds, *i.e.*, which likely share the same topic (see Sec. 4.5.1). Yet, without extra analysis, we cannot

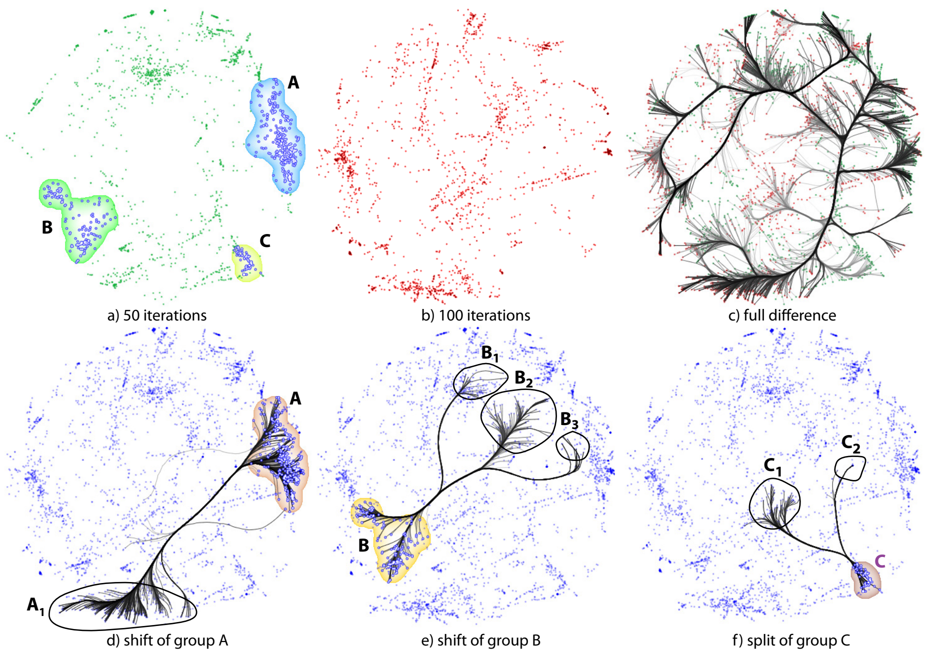


Figure 4.14: Shift between two LSP projections, News dataset, for different numbers of force-directed *iterations*.

relate the two projections, *e.g.*, find out (a) if points markedly change places due to our parameter change; (b) which groups in one projection map to groups in the other projection; and (c) if points in a group in one projection also belong to a group in the other projection.

We answer question (a) by the projection comparison view (Sec. 4.4.7). The result (Fig. 4.14 (c)) shows that there are many quite large point-shifts between the two projections; the bundles' iuntersections also show that groups change locations in the projection. This is a first hint that LSP is not visually stable concerning its number-of-iterations parameter. Next, we manually select three of the most salient visual point-groups in one projection, shown in Fig. 4.14 (a) by the shaded cushions A,B,C. We analyze these in turn. In Fig. 4.14 (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 place in the projection). The same happens for group B (Fig. 4.14 (e)). Yet, the parameter change spreads B more than A – in image (e), we see that B maps to three groups, $B_1..B_3$. These views thus answer question (b). Group C behaves differently (Fig. 4.14 (f)): when we change our parameter, C splits into two smaller groups C_1 and C_2. For question (c), thus, the answer is partially negative: not all groups are preserved in terms of spatial coherence upon parameter change.

4.6 DISCUSSION

Our visualization techniques are implemented in C++ using OpenGL 1.1, and tested on Linux, Windows, and OSX. Next we discuss several aspects of our proposed visualization techniques.

Computational scalability: Overall, we achieve interactive querying and rendering of our views for projections up to 10K points on a commodity PC. In detail: For Delaunay triangulation and nearest-neighbor searches, we use the Triangle [203] and ANN [145] libraries, respectively, which are well-known high-quality tools for the respective tasks. Both libraries can handle triangulations, respective nearest-neighbor searches, in over 100K points in subsecond time on a commodity PC. We accelerate imaging operations using GPU techniques: For distance transforms, we use [32], which is linear in the number of sites and image pixels too. On an Nvidia GT 330M, this computes shaded cushions and performs our Shepard interpolation at interactive frame rates for views of 1024^2 pixels. For edge bundling, we re-implemented KDEEB [93] fully on NVIDIA's CUDA platform. This yields a speed-up of over 30 times (on average) as compared to the C# implementation in [93], and allows bundling graphs of tens of thousands of edges in roughly one second.

Visual scalability: Our image-based visualizations can handle thousands of data points, even when little screen space is available. Clutter, as caused by overlaps of primitives being drawn for the projected points, is reduced by construction, as explained earlier in Sec. 4.4. Apart from that, our proposed techniques have a multiscale property: The parameters α and β (Eqns. 4.5, 4.6) specify 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. 4.1). The bundled views (Sec. 4.4.5) also work in a multiscale fashion, given the inherent property of bundled edge layouts to emphasize coarse-scale connectivity patterns in a graph.

Genericity: Our visualizations can be applied to any DR method, as long as one can compute an error distance matrix telling how much 2D distances deviate from their nD counterparts (Eqn. 2.6). 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 nD coordinates.

Ease of use: Our views use three parameters: α sets the scale of the visual outliers we want to show; β sets how far around a point we want to show information, *i.e.*, controls the degree of space-filling of the resulting visualizations; ϕ sets the percentage of most important missing neighbors we want to show. These parameters, as well as the interaction for selecting point groups

(Sec. 4.4.6) are freely controllable by users by means of sliders and point-and-click operations. Moreover, their behavior is continuous: Small changes in any of the above-mentioned parameters causes small changes in the resulting visualization. This is a desirable outcome that allows users to fine-tune the respective parameters in an easier way.

Comparison: Like Van der Maaten *et al.* [226], we use multiple views to show the same data points to explain a projection, *e.g.*, the false neighbors, missing neighbors view, missing neighbors finder, and group-related maps. However, a difference is that the multiple views in [226] 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. Separately, our views do not need to be simultaneously present (and linked) on the screen –rather, we use these views consecutively, to explore the errors, and their causes, in a top-down fashion (see Sec. 4.4.8). Arguably, this is an easier-to-use design, as a single view is to be studied at a time by its users. Similar to Aupetit [10], our error metrics encode differences in distances in $\mathbb{R}^n$ *vs* $\mathbb{R}^2$. Yet, our error metrics are different. More importantly, our visualizations are also 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 C^∞ rather than constant C^{-1} per-cell interpolation (Sec. 4.4.3). Secondly, our missing neighbors finder (Sec. 4.4.5) shows one-to-many and many-to-many error relationships, whereas all other methods that we are aware in the same class can only show one-to-one relationships. Finally, we also show errors at visual-group level, whereas the other studied techniques confine themselves to showing errors at point-level only.

Let us also note that our projection comparison view (Sec. 4.4.7) is technically related to the method of Turkay *et al.*, which connects two 2D scatterplots to each other by lines linking their corresponding points [224]. Yet, and important difference is that Turkay *et al.* note that line correspondences only work for a *small* number of points. In contrast, we use bundles to (a) show up to (tens of) thousands of correspondences, colored and blended to encode correspondence importance. As such, we argue that our method is technically more scalable, and producing less clutter, than Turkay *et al.*

Discoveries: One potential drawback to mention is that our findings are, from an end-user perspective, limited, as we could not so far tell which of the studied DR algorithms are optimal. Yet, it should be stressed that this was not the aim of our work: Our goal was to introduce a set of visualizations that *help* analyze the effect of parameters on projection quality for several DR techniques. 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. When such a concrete context is given, our tools can be 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 remarks apply to the datasets we used. Our analyzes involving these should be seen purely as test cases for assessing the quality problems of DR projections, and not as findings that solve the underlying goals related to these datasets. As such, globally speaking, our work here is technique-driven, in terms of providing a set of generic tools to be used with care withing specific application contexts, rather than a set of pre-canned solutions for a (very) specific problem.

Limitations: Our views can show (a) which projection areas suffer from low quality; and (b) how two projections differ in terms of neighborhood preservation. Yet, we cannot directly explain (c) *why* a certain DR method decided to place a certain point in some position; and (d) how the user should *tune* (if at all possible) the algorithm's parameters to avoid errors in a given area. That is, we can explain the function $f : P$ (Eqn. 4.1) and its first derivatives over P, but not the inverse f^{-1}. Doing the latter is a much more challenging task – currently not solved by any technique we know of. Explaining such second-order effects to help users locally fine-tune a projection is subject to future work –and, arguably, a highly application-specific work. Separately, the parameter space P of many DR algorithms (or hyper-parameter space, to use a machine-learning terminology) is high-dimensional. So far, we can only analyze the variation of one or two parameters at a time. Extending this to several parameters, and showing the results thereof in an easy-to-comprehend manner, is a second challenging next topic.

4.7 CONCLUSIONS

In this chapter, we have presented a set of visualization techniques for the analysis of quality of dimensionality-reduction (DR) algorithms, in terms of exploring the parameter settings of a generalized projection function. In detail, by generically modeling such techniques as functions from nD to 2D in terms of their distance-preservation error, we proposed several views that help understanding the distribution of false neighbors, missing neighbors, and aggregated projection-errors at both individual point and point-group level. We use several dense-pixel, image-based, visually scalable, techniques such as scattered point interpolation and bundled edges to make our methods visually and computationally scalable to large datasets and also work in a multiscale fashion, in order to emphasize details at a user-chosen level of dtail. We demonstrated our techniques by analyzing the errors caused by the change of various 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 in a lumped way, we propose more local tools to examine how neighborhoods and groups are mapped in a 2D 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 projection details or the exact role of each parameter in the projections.

Future work can target several directions. First, we plan to support 'what if' scenarios, *i.e.*, help users to decide how they could correct local projection problems by shifting wrongly-placed points while dynamically assessing the ensuing overall projection errors. Secondly, we plan to explicitly visualize the *reasons* that determine point placement, *i.e.*, depict the nD variable values which cause points to be placed close to, or far away from, each other. Additionally, we intend to provide tools for local evaluation of projections customized for specific target audiences. By this, we hope to make the operation of DR algorithms more transparent and understandable for users ranging from algorithm designers to end-users.

5 EXPLAINING 3D DIMENSIONALITY REDUCTION PLOTS

ABSTRACT: *2D projections are well known representations to show dimensionality reduction results in the literature, but few works has been done towards the interpretation of the 3D visual representation. Understanding 3D projections created by dimensionality reduction (DR) from high-variate datasets is very challenging. In particular, classical 3D scatterplots used to display such projections do not explicitly show the relations between the projected points, the viewpoint used to visualize the projection, and the original data variables. To explore and explain such relations, we propose a set of interactive visualization techniques. First, we adapt and enhance biplots to show the data variables in the projected 3D space. Next, we use a set of interactive bar chart legends to show variables which are visible from a given viewpoint, and also assist users to select an optimal viewpoint to examine a desired set of variables. Finally, we propose an interactive viewpoint legend that provides an overview of the information visible in a given 3D projection from all possible viewpoints. Our techniques are simple to implement, and can be applied to any DR technique. We demonstrate our techniques on the exploration of several real-world high-dimensional datasets*[1]

5.1 INTRODUCTION

As we have seen in Chapter 3, projections can be used to effectively map a large high-dimensional dataset into a lower dimension (2D), while similarities and/or neighborhoods of the observations are preserved. This, in turn, enables reasoning about groups of related observations, even in datasets where they do not have any connection, such as the association of certain genres of music to specific kinds of pictures, automatically creating slides shows with sound (Chapter 3). Additionally, such 2D plots can be effectively used as a visual support for mapping the local projection quality, encoded by various projection metrics (Chapter 4).

However useful for the above types of analyses, 2D projections suffer from the main problem that do not preserve all the original information when the observations are projected. We may see which observations are similar (or different), which are groups of related observations, where the projection has created errors, and how large these errors are. However, we do not know *why* projected points are similar, *i.e.*, which variables from the original space that was projected contribute most to bringing points close (or far away) from each other in the pro-

1 The text of this chapter is based on the paper *Explaining three-dimensional dimensionality reduction plots* (D. Coimbra, R. Martins, T. A. T. Neves, A. C. Telea, F. V. Paulovich), Information Visualization Journal, SAGE Publ., DOI:10.1177/ 1473871615600010, 2015. The first two co-authors have had equal major contributions to the publication, and should be seen as joint first authors. Specific parts of this work can be assigned to D. Coimbra, as follows: The key idea of explaining 3D projections (as opposed to 2D projections); the design of the dynamic axis legends and curved biplot axes; the design of the viewpoint widget; the selection and analysis of the *segmentation* and *multifield* datasets; and the analysis of view-dependent projection quality (Sec. 5.4.5).

jection. This is an important hurdle in making projections understandable and usable for the grand public. For instance, typical end users are accustomed to graphs or scatterplots whose axes are labeled to describe the quantities they map. These offer essential cues for reading the graphs. Projections do not come, by default, with such annotations. As such, users will arguably encounter difficulties in explaining the visual patterns they see in the projection (e.g. clusters of densely packed points or outlier points). When these difficulties become too large, they may even block the adoption of projection as a visualization technique [27].

The above challenges are present even more so for 3D projections, i.e. projections which reduce the original high-dimensional dataset to a 3D scatterplot rather than a 2D one. For the 3D case, the additional degree of freedom in the projection space allows a lower projection error (as we will see). In the same time, interpreting a 3D scatterplot whose axes do not have a clear meaning is considerably more challenging than doing the same for a 2D scatterplot. For instance, in the 2D scatterplot case, brushing techniques can provide required insight into the dimension values of points of interest, up to the extent where points do not overlap. Using the same mechanism for explaining 3D scatterplots is considerably harder, as selecting 3D points for brushing is much harder. Separately, using 3D scatterplots involves choosing a viewpoint, which may positively or negatively influence the interpretation of the scatterplot.

We propose a set of interactive explanatory visualization techniques to help users answer the above questions and assertions for 3D DR projections by aiding in course correlations. 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. 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 5.2 presents related work in computing and interactive exploring DR projections. Section 5.3 introduces our explanatory visualizations via a simple dataset. Section 5.4 illustrate how our visualizations can answer several questions on 3D scatterplots created by several DR techniques from real-world datasets. Section 5.5 discusses our techniques. Finally, Section 5.6 concludes the paper.

5.2 RELATED WORK

As outlined in Sec. 2.4.3, multidimensional projection methods aim to map high-dimensional space into a low-dimensional (visual) space, so as to preserve the original information as much as possible. Understanding and/or explaining such projections generally raises some questions, which are next discuss.

5.2.1 *Explaining projections*

Interpreting DR scatterplots is not easy. Refining the questions in Sec. 5.1, we aim to address the following goals:

1. assign a meaning to the m dimensions of the mD projection space in relation to the original n variables;
2. assign a meaning to the inter-point distances in $\mathbb{R}^m$ in relation to the corresponding distances in $\mathbb{R}^n$;
3. find a suitable viewpoint (for 3D projections) that best supports answering specific questions;
4. compare the quality of projections for dimensions $m \in \{2, 3\}$ from the perspective of several given tasks.

Goal 1 can be addressed by *biplots* and their variations [81, 79]. Biplots are the multivariate analogue of scatterplots. Instead of using the scatterplot idea of plotting observations along two orthogonal (Cartesian) axes mapping two variables, biplots approximate the multivariate distribution of a high-dimensional dataset in a few dimensions, typically 2 or 3, by superimposing representations of variable values on representations of the observations themselves. As such, they offer the possibility to easily see relationships between (1) individual observations and (2) observations and their variable values [80]. Graphically, biplots can be seen as a scatterplot generalization, in the sense that they have as many axes as there are variables, and these axes can take any orientation in the display (Figure 5.1). Biplot axes support goal (2) above by showing which are the directions of maximal variation of the original n variables in the m-dimensional projection space.

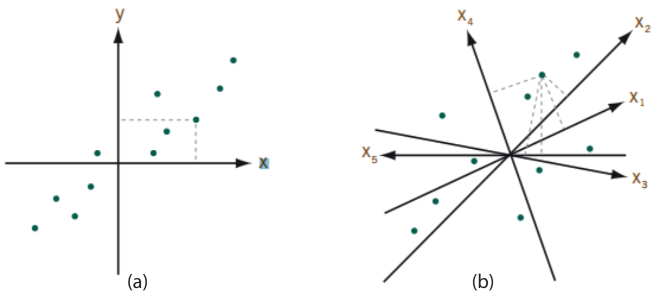


Figure 5.1: Scatterplot (a) and biplot (b) comparison. Green dots represent the instances, or observations, and axes represent variables [81].

Biplots and their axes are usually constructed as follows. 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

$$\mathbf{D} = \mathbf{U}\Delta\mathbf{V}^T \tag{5.1}$$

where $\mathbf{U}$ is a $N \times r$ matrix, Δ 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, $\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Denoting $\mathbf{F} = \mathbf{U}\Delta$, we have $\mathbf{D} = \mathbf{F}\mathbf{V}^T$. The columns of $\mathbf{V}^T$ 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. 5.1 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 $\mathbf{D}$. 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. 5.1, the n unit vectors in $\mathbf{R}^n$. These vectors show the direction of maximal variation in the resulting m-dimensional projection of our n variables [81, 1].

A different approach to goal 1 is given in [28]. Here, a nD 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* [81, 1], are the projections of the nD unit vectors (via Eqn. 5.1) on the two eigenvectors that determine the projection. The x and y axes are annotated with two n-element bar-charts, where the height of each bar shows the contribution of a given variable to the respective axis. 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 data variance not captured by the 2D projection. A similar visualization of loadings is shown in [152].

Goal 2, *i.e.* assigning a meaning to the inter-point distances in $\mathbb{R}^m$ in relation to the corresponding distances in $\mathbb{R}^n$, is addressed by aggregated quality metrics like stress (Eqn. 2.6), correlation [242], neighborhood-preservation plots [163], shape-based metrics [6], and perceptual user studies [130]. While showing the *overall* quality of a projection, aggregate metrics do not show *local* projection errors. 2D distance scatterplots can show the correlation of D^n with D^m [103], but do not show projection problems for *any* point i *vs all* points $j \neq i$. To improve this, Schreck *et al.* compute, for each data point $\mathbf{p}_i$, the projection precision score (*pps*) defined as the normalized distance between the two k-dimensional vectors containing the Euclidean distances between $\mathbf{p}_i$ and its k nearest neighbors in $\mathbb{R}^n$, respectively $\mathbb{R}^2$ [197]. Showing *pps* via a color map helps finding areas where neighborhoods are not preserved. Aupetit proposed several projection-quality metrics for DR techniques [10]: Point stretching and compression metrics show, for each $\mathbf{p}_i$, the aggregated increase, respectively decrease, of distances of $\mathbf{p}_i$'s projection $\mathbf{q}_i$ to other projections $\mathbf{q}_{j \neq i} \in D^m$ *vs* distances of $\mathbf{p}_i$ to all other points $\mathbf{p}_{j \neq i} \in D^2$. Segment stretching and compression show the variation of distances of close point-pairs (i, j) between $\mathbb{R}^n$ and $\mathbb{R}^2$. For a selected point i, the proximity metric maps distances in $\mathbb{R}^n$ from $\mathbf{p}_i$ to all other points $\mathbf{p}_{j \neq i}$ to corresponding points $\mathbf{q}_i \in \mathbb{R}^2$, and thereby shows how (and where) the projection may have distorted the data structure. An overview of quality metrics for multivariate data visualization is given in [20].

Goals 1 and 2 are also addressed jointly by other tools. The early VIBE system allows users to freely place in 2D space several so-called points of interest (POIs), each representing a sample of the nD space under study [153]. 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 nD to 2D. However, this approach requires the user to manually create relevant POIs (samples of the nD 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 [65]. By dragging, pinning, and annotating documents, users can incrementally assign higher-level *semantics* to 2D inter-document distances. The 'dust & magnets' 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 [244]. While we also use interaction to explain a projection, like [65, 244, 153], our focus is to explain projection-space distances in terms of the original nD *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 will not modify the projection, as we consider it to be our 'ground truth', and also give a key role to the nD variables in our explanation.

Goal 3, *i.e.* finding a suitable viewpoint (for 3D projections) that best supports answering specific questions, can be addressed by multiple views, such as three 2D views linked with a 3D scatterplot by interactive selection [176], or interaction and animation, *e.g.* the scatterplot matrix. 'Rolling the dice' 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 [64]. This idea was extended in [194] 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. A similar idea was used by Hurter *et al.* to link 3D and 2D scatterplots [96]. Claessen *et al.* [43] 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.

Goal 4, *i.e.* comparing 2D *vs* 3D DR projections, to find which is more suitable for a specific context (and why), is still an open subject [212]. Several authors argue that 2D DR plots are better for visualizing text documents [149, 237], and that 2D navigation is easier than its 3D counterpart [237]. 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 [201]. 2D DR plots were also found better for search tasks [236] and for tasks involving distance assessment and spatial arrangements [67]. 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 [105]. Dang *et al.* show how 3D glyph stacking can overcome color coding problems in 2D plots [50]. Additional cues such as illumination and depth are proposed in support of using 3D scatterplots [193]. Sanftmann *et al.* argue that high-point densities in scatterplots are better handled by 3D scatterplots [194]. Chan *et al.* argue that 3D projections decrease information loss by allowing better discrimination between data elements [37]. A discussion of contexts where 3D DR projections are preferable to 2D ones is given in [192]. Poco *et al.* compared 2D and 3D DR projections using LSP [163] both quantitatively (by stress metrics) and qualitatively (by controlled user studies) [178]. 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.

Summarizing the above, with needed brevity, we argue that (a) 2D DR plots are generally found more effective for the specific tasks of cluster separation and searching, and require less interaction; while 3D DR plots preserve distances better, but loose appeal due to navigation, orientation, and occlusion problems. As such, we argue that our goal of designing effective interactive exploration tools for 3D DR projections, that keep the benefit of higher 3D projection accuracy as compared to 2D projections, but decrease 3D interpretation costs, is worth investigating.

5.3 EXPLANATORY VISUALIZATIONS

We next detail our interactive visualizations that support the explanatory goals in Sec. 5.2.1. As running example, we use a dataset containing 2814 points, each representing the abstract of a scientific paper (dataset ALL in [158]). From the abstracts, a 9-dimensional feature space was created by removing stop-words and using stemming. Feature coordinates were computed by the term-frequency-inverse-document-frequency count [189]. From this dataset, a 3D projection was created using LAMP [103]. A tenth attribute, not used in the projection, indicates the class of each document, established manually based on the perceived topic of each document.

Figure 5.2 shows the 3D projection using a scatterplot, with points colored by their class attribute. Apart from seeing a few separated point clusters, which seem to capture the class attribute, this image does not tell us more: We do not know how variable values vary along the 3D space; or whether they correlate

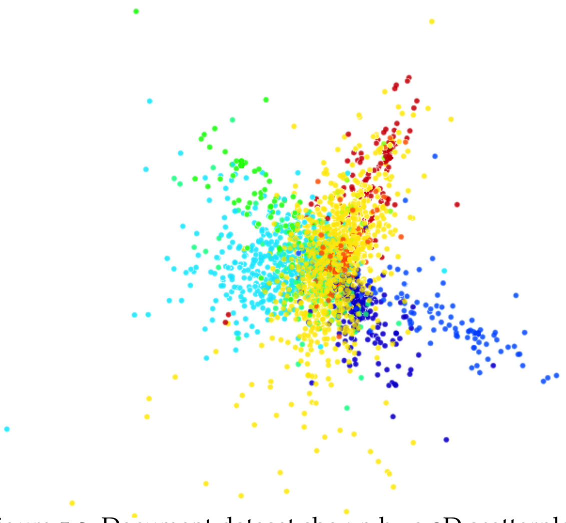


Figure 5.2: Document dataset shown by a 3D scatterplot.

with the clusters or with each other; or how to choose a good viewpoint to examine the dataset. We next show how to answer such questions.

5.3.1 *Enhanced biplot axes*

Standard biplots project the n variables into biplot axes in the low-dimensional mD space using SVD (Eqn. 5.1). This has several problems. First, this assumes that DR is done using a uniform *and* linear transformation. This is not true for nonlinear DR techniques or techniques based on different local projection schemes (Sec. 5.2). Secondly, this assumes that we know the internals of the DR method, such as the SVD matrices $\mathbf{U}$, Δ, and $\mathbf{V}$ (Sec. 5.2.1). Finally, such biplots cannot show the direction and (nonlinear) scaling of the n variables.

We address these issues as follows. For each nD variable i, we create a set of $S = 100$ sample points $\mathbf{p}^i_{1 \leqslant j \leqslant S}$, spread uniformly between the minimum and maximum of variable i in D^n; for the other variables $k \neq i$, $\mathbf{p}^k_j$ take values equal to the average of variable k in D^n. Next, we use the DR projection f (Eqn. 4.1) as a *black box* to project the points $\mathbf{p}^i_j$ to $\mathbf{q}^i_j \in \mathbb{R}^m$, and draw a curve (biplot axis) $c_i = \{\mathbf{q}^i_j\}$ to connect all projected points. Figure 5.3 a shows this. We see how and where the nD axes get mapped in the 3D space. The lengths and bends of the curves c_i tell us about the spread, respectively non-linearity, of the projection. Straight long 3D curves, *e.g.* axes 6 and 8, show variables which are dominant (in terms of data variation) *and* well preserved (in terms of linearity) by the projection. Curved axes, *e.g.* 3 and 1, show (local) non-linearities of the projection. Short axes show dimensions along which data has less variation. Axes intersect in the projection centroid. Figure 5.3 b shows the same dataset, projected with the FBDR force-based scheme in [214]. The extent and overall shape of the resulting 3D point cloud is very similar to the LAMP projection in Fig. 5.3 a. Yet, axes are now significantly more curved and entangled. This tells us directly that FBDR is less good than LAMP if we want to be able to 'read' our nD variables along clearly-separated directions in the projection space.

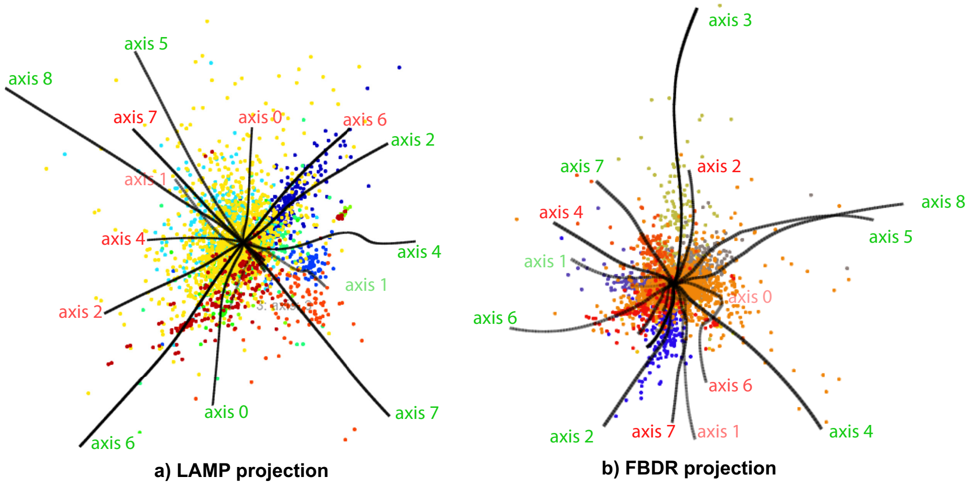


Figure 5.3: Adding curved biplot axes to the 3D projection in Fig. 5.2.

Besides showing the spatial deformation caused by the projection, we can adapt our biplot axes to show the (non)linear nature of the projection. For this, we add labels and ticks to equal-value intervals (in the high dimensional space) to the biplot axes. By looking at the distribution of these labels and ticks along an axis, we can get an idea of the local compression and/or stretching that may have been caused by the projection (see Fig 5.4).

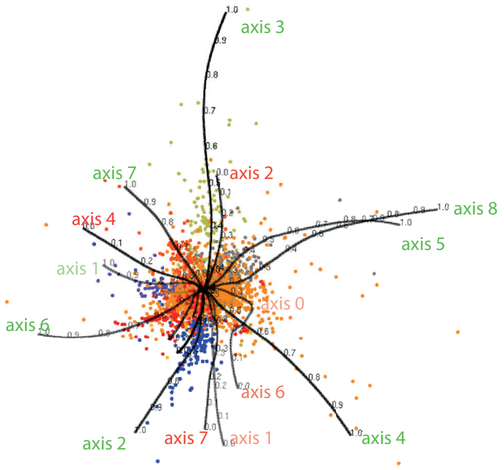


Figure 5.4: FBDR projection with labels and ticks.

5.3.2 *Enhanced axis legends*

Users can view 3D projections from any viewpoint, using a virtual trackball to rotate, translate, and zoom the camera. For such a viewpoint, we denote the screen axes x and y by $\mathbf{x}_1$ and $\mathbf{x}_2$, and the view direction by $\mathbf{x}_3$. Given such a viewpoint, a key user question is "what can I see from here?" This question can be rephrased as: the variations of which original nD variables can we see best along screen axes $\mathbf{x}_1$ and $\mathbf{x}_2$? And which variables cannot that viewpoint show, because they get mapped along the view direction $\mathbf{x}_3$? Consider the analogy with the display of a simple 2D scatterplot of two variables, something that arguably most users are familiar with: The meaning of the screen x and y axes is clear – each such axis maps one of the two input variables. This is not so for our context, since (a) DR projects map many ($n \gg 2$) variables to 3 axes, so an axis will represent a 'mix' of several variables; and (b) we can freely choose any 3D viewpoint to look at the 3D DR projection. We propose to jointly address both (a) and (b), as follows.

Construction: To explain the screen axes, we use three bar charts, or axis legends (Fig. 5.5), one for each of the axes $\mathbf{x}_j$, each having n bars for the n input variables. The height of the i^{th} bar in the legend of $\mathbf{x}_j$ tells how much axis $\mathbf{x}_j$ shows the variation of the i^{th} variable, and is given by the absolute value of

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

Here, $\|c_i\|$ is the length of the (curved) biplot axis c_i, computed as in Sec. 5.3.1. The first term in Eqn. 5.2 is the projected length of c_i on screen axis $\mathbf{x}_j$. High values hereof tell that we can easily see the spread of variable i along axis $\mathbf{x}_j$. The second term in Eqn. 5.2 encodes the linearity of c_i. High values hereof tell that the projection maps variable i to a straight line in the 3D projection space. Low values tell that variable i maps to a curved axis – so reading this variable along the *straight* screen-axis $\mathbf{x}_j$ will be difficult. High values of h_i^1 or h_i^2, *i.e.* long bars in the $\mathbf{x}_1$ or $\mathbf{x}_2$ charts, are desirable, as they tell that the x or y screen axes can be used to directly read the variation of variable i. High values of h_i^3 are undesirable, as they tell that variable i spreads mostly along the view direction, thus it is not observable from the current viewpoint. Hence, we call the $\mathbf{x}_3$ chart: the *observability legend*. We also orient the bars of the x and y legends upwards, and the bars of the observability legend downwards respectively (see Fig. 5.5). This way, the upwards-pointing direction of bars uniformly represents observability (of a variable) in all three legends.

The sign of h_i^j tells if variable i is mapped in the positive or negative direction of screen axis $\mathbf{x}_j$. We show this by a green ($h_i^j > 0$), red ($h_i^j < 0$), respectively gray ($h_i^j = 0$) box under each bar in the axis legends $\mathbf{x}_1$ and $\mathbf{x}_2$. This shows how a variable increases or decreases along a screen axis. For the view-direction axis $\mathbf{x}_3$, we do not show this sign, since data variations along this axis are, by definition, not visible from the current viewpoint. Bars in all three charts are colored to show the identity of the variables by a categorical colormap created with ColorBrewer [26] and labeled by variable names (more about this next).

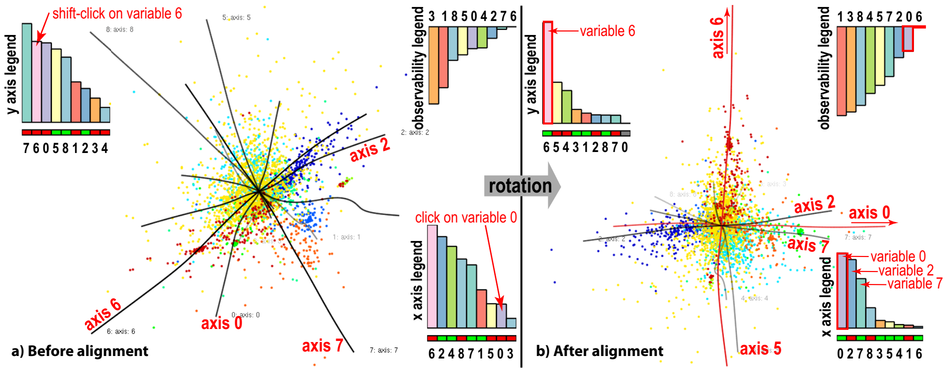


Figure 5.5: Axis legends. Two clicks in the left view will align variables 0 and 6 with the screen x and y axes respectively, leading to the right view.

Sorting legends: We provide two modes to sort legend bars left-to-right. The first mode sorts bars alphabetically on their variable names, so bars for the same variable i appear at the same position in all three legends. This allows one to quickly visually scan and correlate the three legends to see how a given *variable* of interest is visible from the current viewpoint, *i.e.*, answer the question "Along which screen axis (x or y) can I best see this variable?" The second mode sorts bars in decreasing order of their $|h_i^j|$ values. This allows one to quickly see which are the best-visible variables along a given screen *axis*, or answer the question "What does this screen axis show?". In this mode (see Fig. 5.5), bars for the same variable i may not appear at the same position in the three legends, but still have the same color and labels, to help correlation. This mode also addresses the case when we have a high-dimensional dataset, *i.e.*, n is large (tens or hundreds). Since legends are sorted, the most-visible variables along the x and y screen axes are always the leftmost (and longest) bars of the x and y legends. If n exceeds a fixed preset $n_{max} = 20$, we only draw the first n_{max} longest bars. This ensures (a) that the $\mathbf{x}_1$ and $\mathbf{x}_1$ legends always show the n_{max} most-visible variables from the current viewpoint, and (b) that bars are wide enough for their color and label annotations to be readable. For the $\mathbf{x}_3$ legend, the drawn bars tell us which are the *worst visible* variables from the current viewpoint. This helps answering the question "Which variables should I not try to analyze from the current viewpoint?" Summarizing, even when $n > n_{max}$, our three legends can tell us which are the best and worst visible n_{max} variables from any viewpoint. Apart from color coding, bars in all three legends are linked, in both sorting modes, by brushing, similarly to the design proposed in [28]: Whenever one moves the mouse pointer in a bar in a legend, this bar and the two other corresponding bars in the other two legends are highlighted. This way, one can quickly see how important a given variable of interest is along both x and y axes, and also how much of the variation of this variable cannot be observed from the current viewpoint, since it occurs along the view direction.

Linked views: We next use interactivity to support several exploration tasks. As the user changes the viewpoint, *e.g.* by rotating the virtual trackball, axis legends dynamically change, so that one interactively sees how the viewpoint change affects what is mapped along the screen axes. Separately, we set the transparencies of the biplot axes c_i to the values $|h_i^3|$. Axes for variables with low $|h_i^3|$ values get emphasized (opaque), telling that their variables can be well read from the current viewpoint – see *e.g.* axes 7,6,2 in Fig. 5.5a. Conversely, axes for variables with high $|h_i^3|$ values are more transparent, telling that these variables are hard to read from the current viewpoint – see *e.g.* axis 5 in Fig. 5.5b.

Viewpoint selection: We further assist users to choose a good viewpoint by interactive-and-iterative axis alignment, as follows. Clicking any bar i in the $\mathbf{x}_1$ or $\mathbf{x}_2$ legends smoothly rotates the viewpoint to a new one where the biplot axis c_i for the clicked variable is best aligned, *i.e.* has a maximal h_i^j value, with the clicked screen axis $\mathbf{x}_1$ or $\mathbf{x}_2$. Shift-clicking a second bar i' in the other legend (say, $\mathbf{x}_2$, if $\mathbf{x}_1$ was the first click) aligns variable i' with $\mathbf{x}_2$, but constrains viewpoint rotation around $\mathbf{x}_1$. This way, we get a viewpoint which best encodes the variation of two user-chosen variables – *i.e.*, creates the best-possible scatterplot i *vs* i' allowed by the given DR projection – with only two clicks. Figure 5.5 illustrates this by showing how we align variables 0 and 6 with the screen x and y axes (Fig. 5.5a). The resulting alignment (Fig. 5.5b) also shows that axes 0 and 6 (marked red) are slightly curved, so that the projection is non-linear. The y legend shows that the vertical data spread is mainly explained by variable 6. The x legend shows that the x spread is mainly explained by a mix of variables 0, 2, and 7, since the three longest bars in this legend have quite similar sizes. Since variable 0 is best aligned with the x axis, by the alignment procedure, it means that variables 2 and 7 must *also* be well aligned with x too. It thus follows that variables 0, 2, and 7 project to (near) parallel axes in 3D, *i.e.*, they are strongly correlated. To check this, we brush the respective bars in the x plot, which highlights their biplot axes in 3D (apart from highlighting the corresponding bars in the three legends, as explained earlier). As shown in Fig. 5.5b, these are indeed correlated (the respective biplot axes are nearly parallel). Note that our x or y alignment tool is crucial for discovering correlations. Indeed, for the arbitrary viewpoint in Fig. 5.5a, the y bars for *e.g.* variables 7 and 6 are quite similar in length; yet, after alignment, we clearly see that variable 6 is orthogonal to variable 7.

Our approach is related to the legends in [28], which show the variation of the nD variables along the screen x and y axes, and the variation in the view direction (thus, not visible from a given viewpoint). Yet, important differences exist. First, the legends in [28] are static, as their 2D projection is *predefined* by the SVD's two largest eigenvectors. Our dynamic legends help reading the nD variables from a interactively *user-chosen* viewpoint in 3D. For example, the x and y legends in Fig. 5.5a show that that viewpoint does not clearly let us read *individual* variables along the x and y screen axes, many bars are long in these legends. After alignment, the legends significantly change (Fig. 5.5b), telling us that x maps mainly a mix of variables 0, 2, 7; and y maps mainly variable 6. We also see this in the observability legend (Fig. 5.5b, top right): Bars for variables 6, 0, 2 and 7 are shortest (in this order), telling that these variables are indeed almost fully captured by the xy screen-space. In contrast, bars for variables 1, 3 and 8 are longest; this indicates that these variables are poorly observable in the xy screen space for the current viewpoint, since they spread mainly in the z direction. Secondly, while [28] orients bars in all three legends upwards, we chose to orient the observability legend bars downwards. This is in line with the fact that long bars in the observability legend are undesirable (they indicate variables we cannot see), while long bars in the x and y legends are desirable (they indicate variables we can see). Thirdly, the computation of our bar heights is different. In [28], these are the so-called 'loadings' of the input n variables *vs* the two eigenvectors used for 2D projection. Computing loadings requires *explicit* knowledge of the DR method f used (SVD, in [28]). In contrast, we treat the DR method as a black box when creating our biplot axes (Sec. 5.3.1), and compute our bar heights separately as a function of the biplot axes' positions given by the current viewpoint (Eqn. 5.2). Hence, our biplot axes can be straight lines or curves, depending on the (non)linearity of f. In contrast, [28], which uses the biplot set-up in [1], assumes a linear projection.

5.3.3 *Viewpoint legend*

Dynamic axis-legends help seeing which variables are visible along the screen axes from a given viewpoint, and also choose a good viewpoint to examine a *given* variable pair. Our next question is: Given a 3D DR projection, which relations (between *all* variable pairs) can we see well if we had time to go through *all* viewpoints?

We answer this question by a new interactive widget: the viewpoint legend (Fig. 5.6). The widget uses a sphere S (Fig. 5.6a); each point $\mathbf{v} \in S$ maps the viewpoint for the view direction $\mathbf{c} - \mathbf{v}$, where $\mathbf{c}$ is the center of S. Thus, S captures all possible viewpoints we can examine our 3D DR projection from. The central cross shows the current viewpoint. We uniformly sample S, in polar coordinates, by 400×400 viewpoints, and define the *quality* of each sample-viewpoint $\mathbf{v}$ in terms of showing the variable-pair $(i, j \neq i)$ as

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

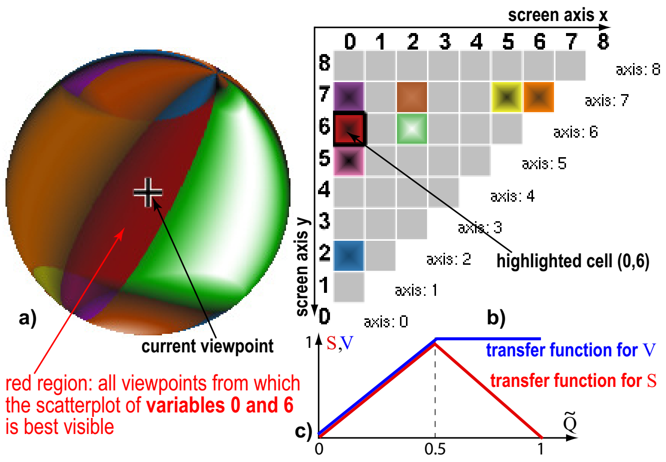


Figure 5.6: Legend for viewpoint shown in Fig. 5.5 right.(a) Viewpoint sphere; (b) Matrix-plot view; (c) Transfer functions for color and luminance of the viewpoint sphere.

Intuitively, q tells how well we can see from $\mathbf{v}$ the variation of variable i *vs* j, modulo all possible rotations of $\mathbf{x}_1$ and $\mathbf{x}_2$ in the view plane around the screen-normal axis $\mathbf{x}_3 = \mathbf{c} - \mathbf{v}$. This depends on how well the DR from $\mathbb{R}^n$ to $\mathbb{R}^3$, *and* the 3D-to-2D (screen) projection given by $\mathbf{v}$, capture this variation. Large $q(\mathbf{v}, i, j)$ values tell that the two biplot axes i, j are large *and* form a large angle (maximally, $90°$) on the view plane. Such viewpoints are interesting to explore, as they show existing independent variable-pairs which also have large spreads.

For each sample viewpoint $\mathbf{v}$, we compute the maximal value of q for all variable-pairs (i, j)

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

For all $\mathbf{v}$, we also compute the normalized maximal quality $\bar{Q}(\mathbf{v}) \in [0, 1] = Q(\mathbf{v}) / \max_{\mathbf{u} \in S} Q(\mathbf{u})$ and the variable-pair $p(\mathbf{v}) = (i, j)$ which maximizes q at $\mathbf{v}$. Here, (i, j) are the variables that define the 2D scatterplot-like view we can best see from viewpoint $\mathbf{v}$. Next, we select the set P of $C = 8$ distinct variable-pairs that have the largest values of q over all viewpoints $\mathbf{v} \in S$. P gives the C variable-pairs we can best visualize from all possible viewpoints. We assign to each pair $p \in P$, thus to all C best-visible variable-pairs, a distinct color $c(p)$, using a categorical colormap, and color the sphere points $\mathbf{v}$ as follows: If $p(\mathbf{v}) \in P$, we use for $\mathbf{v}$ the color $c(p)$, else we use the color gray. Next, we modulate the saturation S and brightness V of the assigned color at $\mathbf{v}$ by the quality $\bar{Q}(\mathbf{v})$ using the transfer functions shown in Fig. 5.6c. This effectively maps $\bar{Q}(\mathbf{v})$ to the shading of the sphere: Low values are dark; mid-range values are saturated; and high values are white. Finally, we render this sphere using standard bilinear color interpolation over a quad mesh defined by our sample points $\mathbf{v}$.

To help interpreting the shaded sphere, we add a separate matrix plot view (Fig. 5.6b). Each variable-pair (i, j) maps to a cell in this plot. Cells are colored using a two-color scheme, as follows. The first color is $c(p)$ for cells of pairs $p \in P$, and is gray for other cells. The second color maps the value $\max_{\mathbf{v} \in S} q(\mathbf{v}, i, j)$ to a gray value between black and white. Cells are colored by linearly interpolating between the first color, assigned to the cell-border, and the second color, assigned to the cell-center. The matrix plot thus shows both the C best visible variable-pairs, encoded by their respective colors, and also the relative quality of different variable-pairs, encoded by the brightness of their respective cell-centers.

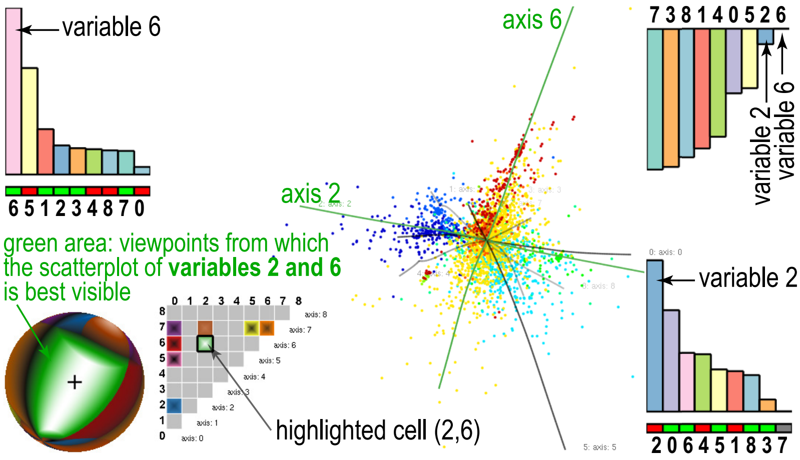


Figure 5.7: Selected viewpoint best showing scatterplot of variables 2 and 6.

Figure 5.7 shows the added-value of our viewpoint-legend and matrix-plot for our documents dataset. We explore the viewpoint space interactively, as follows. Rotating the sphere changes the current viewpoint, which in turn dynamically updates the axis bar-charts (Sec. 5.3.2). Conversely, rotating the 3D scatterplot (either manually or by axis-alignment animation, see Sec. 5.3.2) turns the sphere in sync to show the newly selected viewpoint. The cell for the current viewpoint is highlighted on the matrix plot, so we can directly see which variable-pair is best visible from that viewpoint, *e.g.* $(2, 6)$ in Fig. 5.7. Clicking any cell (i, j) in the matrix plot smoothly rotates the viewpoint to one where the variable-pair (i, j) is best visible, *i.e.* goes to the viewpoint $\mathbf{v}$ where $q(\mathbf{v}, i, j)$ is maximal. This allows quickly navigating to such a viewpoint for any given variable-pair – *i.e.*, constructs the best scatterplot $(i, j), \forall i \neq j$ by one click.

The viewpoint legend helps answering several questions, all related to choosing informative viewpoints for 3D DR projections, as follows:

Where from should I examine pair (i, j)? Large same-hue sphere zones, *e.g.* the green one in Fig. 5.7, show view-space areas from which the variable-pair (i, j)

is best visible. Looking up green in the matrix plot shows that this zone maps the variable-pair $(2, 6)$.

Is there any good viewpoint for (i, j)**?** Small color-zones show that some variable-pairs are hard to see, since only few viewpoints allow that. This tells users not to expect to 'create' such scatterplots from this DR projection, as this is very hard or even not possible. In other words, if understanding the correlation of such variable-pairs is important, one should first change the DR projection.

How easy is it to examine (i, j)**?** Large bright highlights in sphere zones show that the respective variable-pair is easy to examine from many close viewpoints. Given our quality definition (Eqn. 5.3), this means that the spread of values for these variables is large compared to other variables, *and* that the biplot axes' angles for these variables is large. This tells that creating scatterplots for the respective two variables is very easy – just move anywhere in the respective highlight and you'll get the desired scatterplot. Moreover, the matrix-plot cell brightnesses tell us how easy is it to examine their respective variable-pairs from *all* possible viewpoints: Bright cells tell that there is at least one viewpoint from where the respective pairs can be examined well (selectable by clicking that cell); dark cells tell that no such viewpoints exist.

What can I see from a given viewpoint? Highlights show viewpoints from where the variable-pair given by the color around the highlight is best visible. Dark zones on the viewpoint sphere, like the ones just outside the green zone in Figure 5.7, tell that there is no easy-to-see variable-pair when looking at the plot from the corresponding viewpoints. This is so since the pair which is best visible from such viewpoints has a low quality, as indicated by the dark colors. Hence, such zones tell that their respective viewpoints are arguably not useful for *any* visualization task.

How to relate more than 2 variables? Color-zone *borders* show viewpoints where the best visible variable-pair changes for small viewpoint rotations. These are typically bad viewpoints to examine a single variable-pair. However, as we shall see in Sec. 5.4.2, these are good viewpoints to examine *groups* of three or more variables.

5.4 APPLICATIONS

We next use our explanatory visualization techniques (enhanced biplot axes, axis legends, viewpoint legend) to explore 3D DR projections and aid in coarse correlations. They were constructed by three different DR methods, for four different datasets. By showing more datasets, we can easily explain how we address different kinds of questions with our tools, since each one has different data and, consequently, different questions related to it.

5.4.1 *Wine dataset: Finding good DR projections*

This $n = 12$D dataset has 4898 points, each being a different sample of *vinho verde* white wine [46]. Variables include chemical properties, *e.g.* acidity, sugar and sulfur contents, chlorides, density, pH, and alcohol percentage. The last attribute is a user-assigned quality level. Tasks for this dataset involve finding correlations of the first 11 variables on the one hand, and the quality on the other hand, over specific subsets of points; if found, such correlations could be next used to design automatic quality predictors [46]. To use DR for such tasks, we first must decide which DR method is best suited. One way for this is to select the DR method that minimizes aggregated projection errors, also called aggregated stress [141]. Yet, many state-of-the-art DR techniques will yield quite similar error values, so such aggregate errors are not discriminatory enough.

We consider here three DR methods: FBDR [214], ISOMAP [218], and LAMP [103] to project our dataset to 3D (other DR methods can be equally easily used). Figure 5.8 shows the obtained projections. For this dataset, these three projections yield very similar values for the normalized stress metric (Eqn. 2.6): 0.75 (ISOMAP), 0.81 (FBDR), and 0.83 (LAMP). Hence, how to say which DR method is best for discovering variable correlations? Showing our biplot axes helps us here (Fig. 5.8). We see that FBDR and ISOMAP create, overall, quite twisted axes, unlike LAMP. Reading data values and/or finding if such axes are highly correlated (nearly parallel) or independent (nearly orthogonal) is clearly much easier if our axes are *straight* lines rather than curves. Our first finding is, thus, that LAMP is better for variable exploration in general.

However, the above does not imply that LAMP would be the best projection for more specific tasks, like exploring correlations of just two specific variables. Consider, for example, *alcohol* and *acidity*. We see that the *alcohol* axis is comparably straight for FBDR and LAMP – hence, we cannot yet rule out FBDR as a useful projection for this task. To study correlations against *alcohol*, we first click on the *alcohol* bar in the y legend to align it with the screen y axis, in all three plots. Next, we use the same procedure to align *acidity* with the screen x axis (one click on the *acidity* bar, x legend). For extra insight, we also color points by *acidity* values, using a blue-yellow-red divergent colormap. We now get several extra insights: First, we see that the x legend for FBDR has many bars of nearly equal size to *acidity*. Hence, either FBDR does not succeed in separating these variables during projection (which is bad), *or* we just discovered that these variables are highly correlated (which is a good finding). Yet, LAMP shows a clear exponential drop-off of the same bar-lengths. Since LAMP's projection-error is roughly equal to to FBDR's, it means that the respective variables are *not* correlated, hence the lack of separation in FBDR is a limitation of FBDR. Separately, we see that ISOMAP creates a twisted *acidity* axis, and also shows a similar artificial correlation of variable-projections along the x screen axis. Hence, we decide that LAMP is better than ISOMAP. Summarizing all above, we conclude that LAMP is the best of the three projections (LAMP, ISOMAP, FBDR): It has

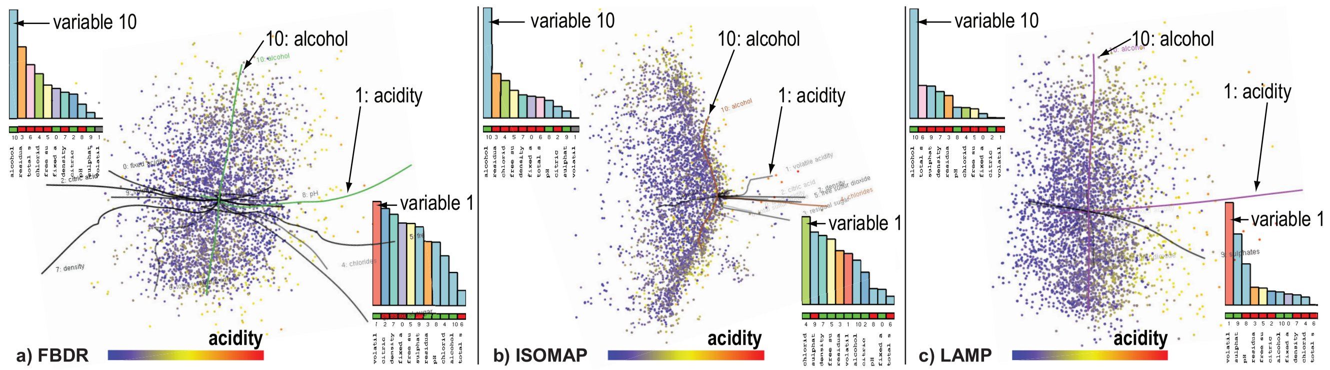


Figure 5.8: Selecting the best projection among three DR techniques (FBDR, ISOMAP, LAMP) using biplot axes and axis legends. See Sec. 5.4.1.

a similar normalized stress metric, but succeeds best in creating straight, and well-separated, variable-axes in 3D projection space.

5.4.2 *Multifield dataset: Explaining projection shapes*

This $n = 10$D dataset, from the IEEE Vis 2008 contest, encodes a time step of a multifield simulation dataset describing the formation of the early Universe [151]. Variables encode matter density, temperature, and concentrations of 8 chemical species at 200K sample points. By freely rotating the 3D DR projection of this dataset(Figure 5.9), done using LAMP, we notice that the projection appears to be locally a 2D saddle-like manifold (point-cloud surface). We next want to better understand the shape of this surface, and find the variables that determine it.

To do this, we turn on our biplot axes. We immediately notice that axis 7 is by far the longest – so variable 7 is important for explaining the projection's shape. Aligning variable 7 with the y screen axis shows that the projection appears to have a 'saddle' shape (Fig. 5.9a). We also see that axis 7 is nearly orthogonal to all other 9 biplot axes. Hence, the y spread of the projection is mainly due to variable 7.

The viewpoint legend in Fig. 5.9a shows next that variable 5 has a large variation which is largely independent on variable 7 (bright green zone on sphere; bright green cell in the matrix plot). To better explore the shape variation due to variables 5 and 7, we next color points by variable 5, via the same colormap as in Fig. 5.8. The result (Fig. 5.9a) shows that the x stretch of our saddle shape is well explained by variable 5, which is high to the left and low to the right, as shown by both the colormap and the red cell under the variable-5 bar in the x legend. In this figure, we also notice an interesting 'spike' line-like outlier in the top-left area. We can explain how this spike, as an specific internal substructure, align with specific axes by looking at them and see that the spike aligns best with axes 5 and 6. Iteratively aligning the x axis (click on variable-5 bar in x legend, then click on variable-6 bar) shows that the spike best aligns with axis 6, as the x bar for variable 6 is largest. Figure 5.9b shows this viewpoint, with points colored by variable 6. We can now easily explain the spike as the locus of points having large variable-6 values (yellow..red). Indeed, all other points (on the saddle shape, not on the spike) have low variable-6 values (blue).

The viewpoint legend in Fig. 5.9b shows that there are many viewpoints from which variables 6 and 7 project as independent axes (large brown area with bright highlight on sphere; bright highlight in the selected matrix-plot cell). Hence, variable 6 is indeed independent on variable 7, which was found the most important for explaining the saddle shape. Aligning variables 6 and 7 with the x and y axes respectively (two clicks in the x and y legends) shows both the spike outlier and the saddle shape in a single view (Fig. 5.9c). This view also shows that axes 5 and 6 are almost parallel, so variables 5 and 6 are highly cor-

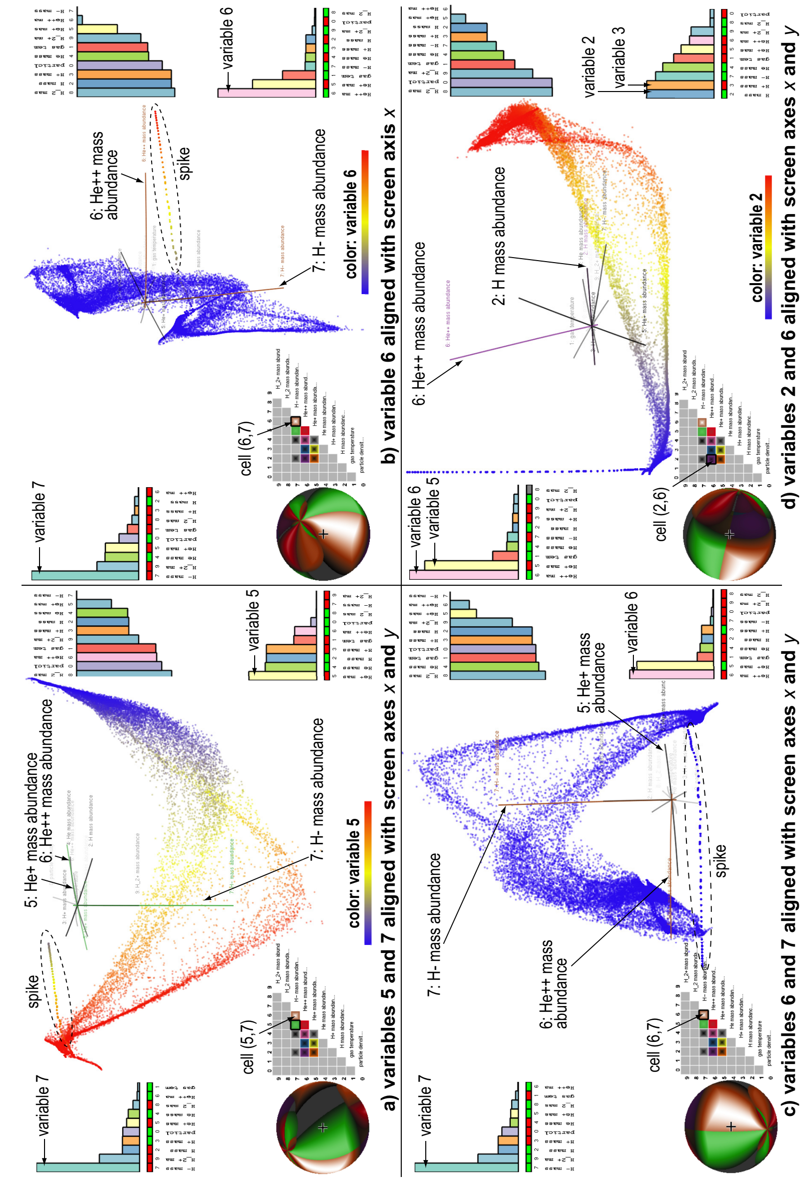


Figure 5.9: Explaining, in terms of variables, the shape of the 3D LAMP projection of 10-variate multifield simulation dataset (see Sec. 5.4.2).

related. We see this also in the viewpoint legend: The current viewpoint, which best shows variables 6 and 7, is very close to the brown-green zone border on the sphere. Also, both brown and green zones have very large bright highlights, *and* the brown-green border is also bright. Hence, most viewpoints which best show variables 6 and 7 *also* best show variables 5 and 7. We thus refine our earlier explanation of the saddle: This shape is best explained by variable 7 (in one direction) and variables 5 *or* 6 (in an orthogonal direction).

To explore variable 6 further, we look at its row in the matrix plot, and click the purple cell, to show its variation against variable 2. This aligns variables 2 and 6 with the x and y axes respectively, yielding the view in Fig. 5.9d. The x and y axis legends show now clearly that variables 5 and 6, respectively 2 and 3, are highly correlated, since they have nearly equal *and* almost maximal bars.

As a final point, let us consider the effort required to explain the spike and saddle shapes present in the 3D scatterplot when using only classical projection exploration tools such as the virtual trackball for rotation and the ability to color all projection points by the values of a chosen variable. Rotating the scatterplot so that we best see the spike outlier, *i.e.* with the spike nicely aligned with the y axis, can take anything between 10 seconds (for an expert user) and 2 to 3 minutes (for someone not familiar with the virtual trackball being used), based on our own observations when using the tool. In contrast, this takes just two clicks on the x and y legends, as explained earlier. Finding that the spike is best explained by variable 6, while the saddle's spread in orthogonal direction to the spike is best explained by variable 2, requires, with standard tools, iteratively selecting each of the 10 variables to color map the projection, detecting visually which is the strongest color gradient aligned with the spike, respectively saddle, and memorizing this value. Using our tools, the color cycling is not required; we can directly see which variables align with specific scatterplot structures in terms of both biplot axes and axis legends.

5.4.3 *Segmentation dataset: Comparing 2D and 3D projections*

Our third dataset has 2300 points with $n = 19$ variables. Each point describes a randomly chosen 3×3 pixel-block from 7 manually segmented outdoor images, using 19 statistical image attributes, such as color mean, standard deviation, and horizontal/vertical contrast [71]. An extra manually-set label attribute, not used in the DR projection, encodes the image type for each point [103, 164]. Tasks for this dataset relate to designing automated image classifiers (using the 19 attributes) to match the manual classification (label attribute) [167].

Figure 5.10 shows this dataset using a 3D DR projection created by LAMP. By freely rotating this projection, with points colored by label values, we see that the longest biplot axis maps variable 0 (*region-centroid-col*). Aligning this axis with the y screen axis (click on *region-centroid-col* bar in the y legend) brings the viewpoint into a large red area on the viewpoint-legend sphere. In the matrix plot, we see that red maps the variable-pair $(0,3)$. We next click this cell to go to the best viewpoint from which we can examine variables 0 and 3 (Fig. 5.10a). The axis legends tell now that y explains almost only variable 0, while x explains mainly variable 3 (*short-line-density*). This viewpoint gives us two other interesting insights. First, we see that variable 0 has almost no correlation with the label-ID, *i.e.*, variable 0 takes virtually all values in its range for *any* single label-ID value. Next, by slightly rotating the viewpoint around the y axis (variable 0), we see that axes 1-18 are located roughly in a plane orthogonal to axis 0. Together, the above tell us that variable 0 is not useful for classification, even though it is the most important in terms of variation; and that the emerging clusters can be explained by variables 1-18.

To better understand the correlation of variables 1-18 with the label-ID, and thus get more insight into developing a classifier, we could next (a) remove variable 0 from the input dataset and redo the 3D DR projection (since we decided that this variable is not interesting); (b) view the current 3D projection from a suitable angle (to ignore the spread along axis 0); or (c) use a 2D DR projection rather than a 3D one (since Fig. 5.10a suggests us that all interesting data variation occurs in a plane).

We examine next option (b). In the matrix plot in Fig. 5.10a, we see that all brightly-colored cells are in columns 0 and 3, *i.e.*, the best viewpoints showing independent variable-pairs always involve variables 0 and 3. The best such viewpoint (brightest red cell) maps variable-pair $(0,3)$ we just studied. We thus now choose to align biplot axis 3 with the y screen axis, and biplot axis 0 (which we are not interested in) with the viewing direction z (Fig. 5.10b). We now see a much clearer segregation of points by label-IDs into separate same-color clusters. This shows that there exist, indeed, correlations of the label-ID with attributes 1-18 – thus, attributes 1-18 hold enough information to design a classifier. The biplot axes in Fig. 5.10b help refining this insight. For instance, we see several strongly-correlated variables: The group of axes pointing downwards (*short-line-density-2*, *hedge-sd*, *hedge-mean*, and *vedge-mean*) all describe image edge features. The group of axes pointing to the left (*raw-red-mean raw-green-mean*, *value-mean*, and *intensity-mean*) all capture means of the image colors. Correlating next these variables with the label variable (point colors) is a first step into explaining the clusters – for instance, we can now easily explain the isolated orange cluster as containing image-blocks having highly saturated colors.

We next examine option (c). For this, we compute a 2D projection using again LAMP. Figure 5.10c shows the result, with points colored again by label-ID. The overall placement of clusters is quite similar, but not identical, to those in the 3D projection in Fig. 5.10b. To see which of these two images is a more faithful

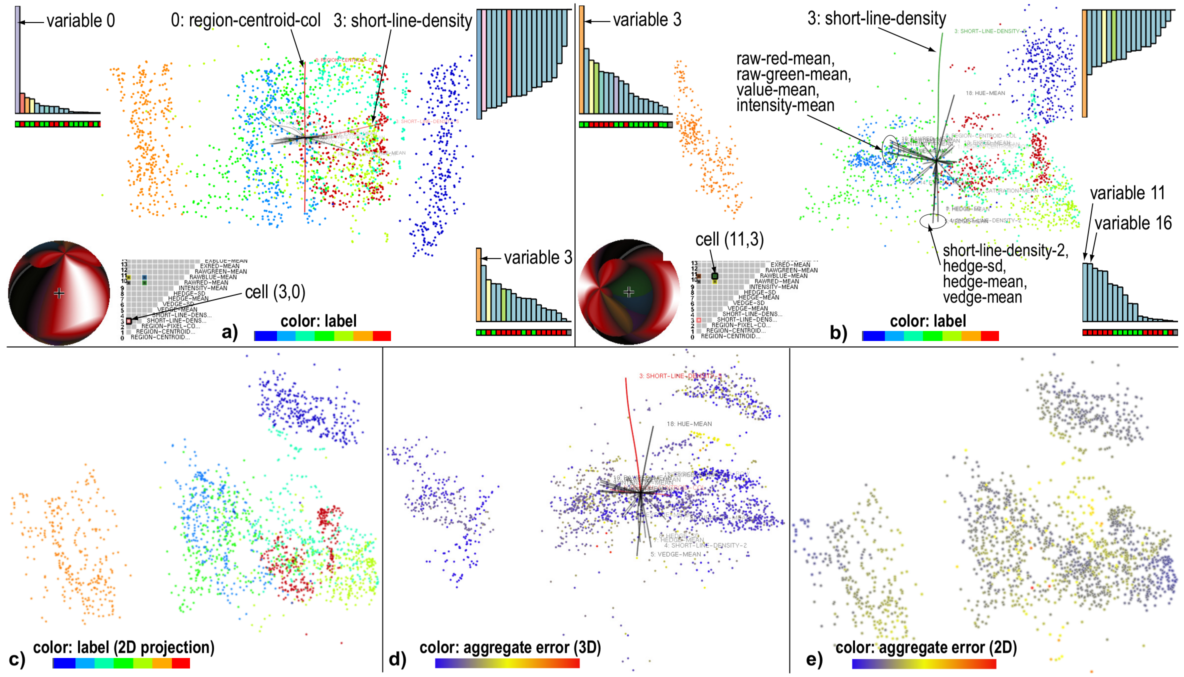


Figure 5.10: Visualization of 19-variate image segmentation dataset using 3D projections (a,b,d) and 2D projections (c,e). See Sec. 5.4.3.

projection, we compute, for each point i, the aggregate normalized projection error $e_i^m \in [0, 1]$

$$e_i^m = \sum_{j \neq i} \left| \frac{d^m(\mathbf{q}_i, \mathbf{q}_j)}{\max_{i,j} d^m(\mathbf{q}_i, \mathbf{q}_j)} - \frac{d^n(\mathbf{p}_i, \mathbf{p}_j)}{\max_{i,j} d^n(\mathbf{p}_i, \mathbf{p}_j)} \right|. \tag{5.5}$$

Here, d^n, d^m, $\mathbf{p}$, and $\mathbf{q}$ have the same meaning as in Eqn. 2.6. The error e_i^m, $m \in \{2, 3\}$, tells how well the mD projection of a point i approximates its placement in $\mathbb{R}^n$ from the perspective of its distances to all other points $j \neq i$. More details on this metric are given in Chapter 4 and [141].

Figures 5.10d and 5.10e show the errors e_i^3 and e_i^2 for the 3D and 2D projections in Figs. 5.10b,c respectively, color mapped as in Fig. 5.9. While both projections 'spread' errors quite uniformly over all points, and do not create any extreme errors, we see that e_i^3 is overall lower than e_i^2. So, the 3D LAMP projection preserves the original nD distances better than 2D LAMP. Hence, for this dataset, using a 3D DR projection, with a suitably chosen viewpoint provided by our exploratory tools (Fig. 5.10b) is better than using a 2D DR projection generated by the same DR technique. This is not entirely surprising, once we understand Fig. 5.10a: The 2D projection has to accommodate the large variation of variable 0 in the same (limited) 2D space used to project all other 18 variables. In contrast, the 3D projection can freely spread all this variation along a separate spatial dimension. Thus, examining the 3D projection from the *single* viewpoint shown in Fig 5.10b – which is roughly equivalent to a 2D projection of variables 1-18 – is better, error-wise, than using a 2D projection of the entire dataset.

Note that the use of our explanatory tools is very different in this use-case than in the one discussed in Sec. 5.4.2. Indeed, in Sec. 5.4.2 we used our tools to select a *variety* of viewpoints, which next helped us explain the projection's shape in terms of variables. In the example here, we used our tools to decide that we can best explore the projection from a *single* viewpoint, and next to choose this viewpoint.

5.4.4 *Software dataset: Finding meaningful clusters*

Our fourth and final example uses a set of 6733 open-source software projects written in C. The source code of each project was downloaded to compute 11 code quality metrics as averages over the project's code files. A 12th metric gives the number of downloads of each project [142]. This yields a $n = 12$D dataset with 6733 points. While [142] explored the statistical correlation of project quality with download count, we want to get finer-grained insights of the types of projects involved in the studied code-base collection.

For this, we use a 3D LAMP projection of our 12D dataset (Figs. 5.11 a-c). We first find the best-visible variable-pair from any 3D viewpoint, by clicking the bright green cell in the matrix-plot in Fig. 5.11a. This gives us variables 2 (*ln-cof*, or average coupling-factor, *i.e.* the number of function-calls between files [123]) and 7 (*ln-sum-tloc*, or total number of lines-of-code). Next, we align axis 2, the longest of these two biplot axes, with screen x axis (Fig. 5.11a). We notice two well-separated point clusters (A, B), which spread orthogonally to biplot axis 2 (*ln-cof*). To understand what these mean, we color points by variable 2. This shows that clusters A and B contain points having two different ranges of *ln-cof* values: A contains low-coupling systems (such as libraries), while B contains medium-coupling systems (such as full applications). We also see here a third cluster (C) formed by very high *ln-cof* points. These points are also orthogonal to axis 7 (*ln-sum-tloc*). Hence, to check if variable 7 explains cluster C, we next color points by variable 7 (Fig. 5.11c): We now indeed see that nearly all points in C have low values of variable 7, and all points in A and B have high values for variable 7. Thus, cluster C contains highly-coupled, small-scale software systems (small applications). Summarizing, we found that our 3D DR projection groups our 6733 software projects in 3 classes: large software projects (high values for *ln-sum-tloc*), further split by project type into libraries (A) and full applications (B); and C, containing small applications (low values for *ln-sum-tloc*). The entire 3D analysis requires just three clicks: one to align the screen x and y axes with the best-separated variables *ln-cof* and *ln-sum-tloc*; and two further clicks to color points by values of these variables respectively.

As for the segmentation dataset (Sec. 5.4.3), we want next to see if a 2D DR projection could give us the same insight given by our 3D DR projection, *i.e.* that our 6733 software projects can be grouped in 3 distinct classes. For this, we first color our 3D-projection points by their aggregated projection error e_i^3 (Eqn. 5.5). Figure 5.11c shows this. Next, we do a nD-to-2D projection (also by LAMP), and color it by its projection error e_i^2 (Fig. 5.11d). Comparing Figs. 5.11c and d, we see that, like for our segmentation dataset, both e_i^3 and e_i^2 are uniformly spread over their respective projections, with $e_i^3 < e_i^2$ on average. However, in contrast to the segmentation dataset, we see that the 3D DR projection creates three clusters (explained by variables *ln-cof* and *ln-sum-tloc*, as discussed); the 2D projection creates only two clusters A' and B' (Fig. 5.11d). By manual brushing of the displayed data points, we found that A' contains a mix of points in A and B (large libraries *and* applications), while B' roughly corresponds to C (small systems). This is also visible in Figs. 5.11a,b: Rotating the viewpoint along the view sphere, and looking at the variation of the axes legends (or alternatively, at the biplot axes), we find no viewpoint in which $n-1$ axes reside in, or close to, a plane. Thus, three projection-dimensions are truly needed to show the data variation that encodes the three clusters – *i.e.*, we need a 3D DR projection to obtain a view that segregates our software systems into three clusters corresponding to large libraries, large applications, and small systems. A 2D DR projection can only segregate software projects into large and small systems, but not segregate based on the coupling type (applications *vs* libraries).

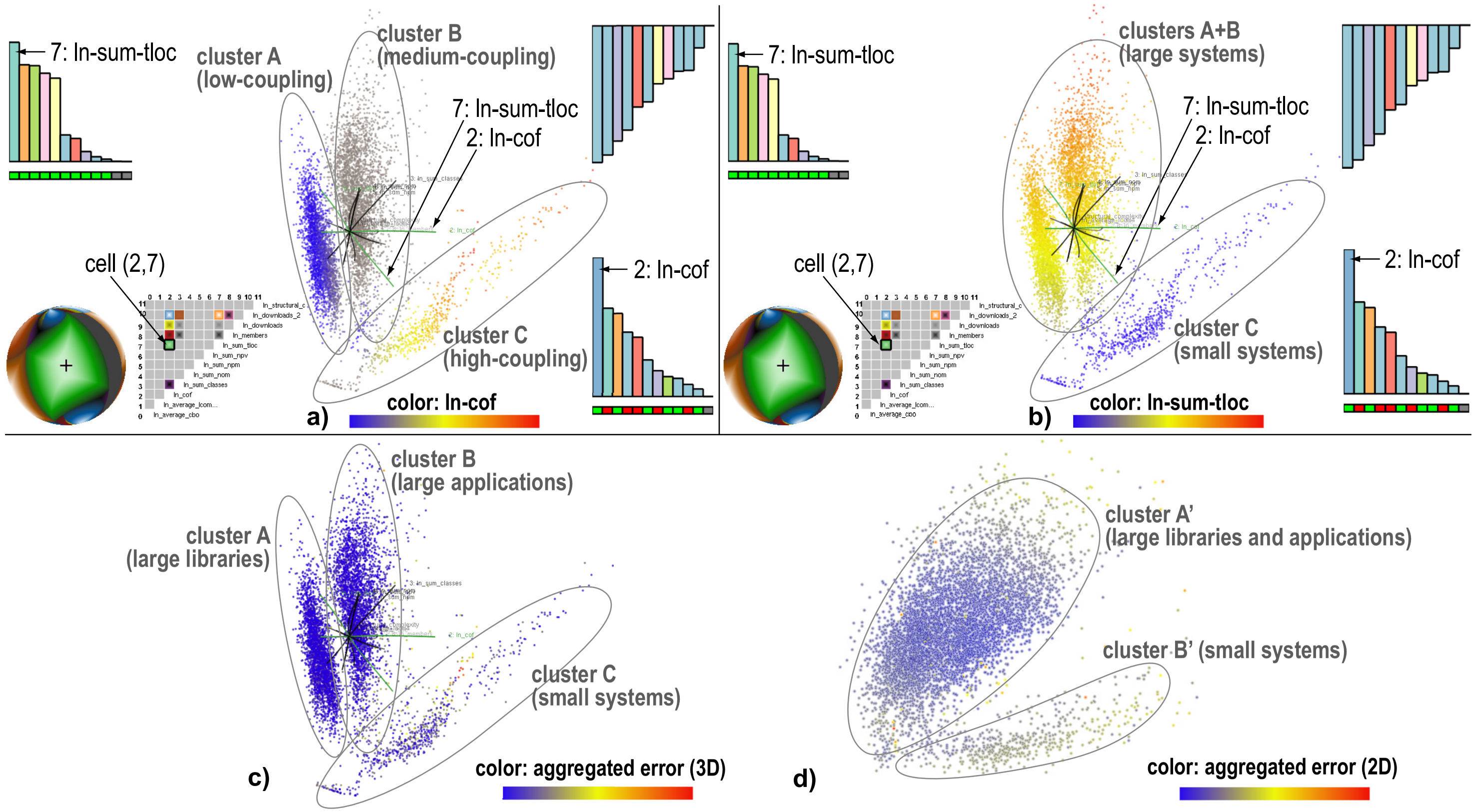


Figure 5.11: Visualization of 12-variate software metrics dataset using 3D DR projections (a-c) and corresponding 2D DR projection (d). See Sec. 5.4.4.

5.4.5 *Axis alignment* vs *projection quality*

The key added-value of our explanatory tools is related to cases when one chooses to use 3D, rather than the more commonly-used 2D, DR projections. Indeed, for 3D projections viewpoint and navigation choices critically affect obtained insights [163, 201]. Let us explain this. Our final 2D view which is rendered on the screen can be seen as being created by 'concatenating' a nD-to-3D DR projection (P_{n3}) with a 3D-to-2D screen projection (P_{32}). As mentioned, P_{n3} typically has a lower error than a direct nD-to-2D DR projection (P_{n2}). Our tools allow understanding and controlling the error given by P_{32}; in contrast, a P_{n2} does not allow any kind of similar error control. For instance, we can interactively change P_{32} to select which variables, or dataset parts, are finally best visible. If, in any view, one axis is small, it means that this variable is not visible in that view, so we cannot reason about it. However, we can rotate the view by aligning this axis to the 2D screen and next interpret the resulting view, thereby obtaining the best view that shows the spread of that variable. In particular, a P_{32} using the two longest axes is as precise as a direct P_{n2}, in terms of stress error. Note that this cannot be done with a direct P_{n2}: If an axis is small in such a projection, we cannot do *anything* to interactively improve that, and no interpretation of data-variations along that axis is possible. Regarding occlusion, our solution is as good as, or better than, using a direct P_{n2}: In our 2D views, occlusion means that 3D points overlap along view lines; yet, we can choose other 3D viewpoints where such overlaps are decreased; in contrast, such overlaps occur in any P_{n2} too, and we cannot do anything to decrease them in that case.

Studying the relationship of the quality of a 3D projection as compared with a 2D projection of the same data is worth a more detailed study. One important question related to this is: How does the quality of a 3D projection changing when the user changes the viewpoint to examine it, *e.g.*, when aligning the projection along, or orthogonally to, the longest or shortest biplot axis? To answer this question, we performed several experiments, described next.

In the first experiment, we created a synthetic dataset having four variables and 1000 observations. The advantage of using a synthetic dataset is that we can precisely control the distribution of observations in high-dimensional space and, thereby, control the sizes of the emerging biplot axes. The dataset was next projected to three dimensions using LAMP. Using this dataset, we want to analyze how the projection quality changes when varying the viewpoint between the position where the longest biplot axis is best viewed (*i.e.*, the biplot axis is parallel to one of the screen axes), and the position where this axis is worst viewed (*i.e.*, the biplot axis is orthogonal to the screen). To ensure that our dataset indeed has one long biplot axis, and knowing that the size of a biplot axis corresponds to its variable variance, we artificially generated three variables with similar pseudo-Gaussian distributions with 5000 as mean and 1500 as standard deviation. This delivers us three corresponding biplot axes of similar lengths. The fourth (longest) axis was created by distributing variable values using an exponential function $f(x) = x^{1.5}$ where $1 \leqslant x < 1000$. Figure 5.12 shows the vertical rotation of the long biplot axis (variable '3:exp'), from the best position where this axis is aligned with the y screen axis (Fig. 5.12a) to the worst position, where the axis is orthogonal to the screen.

To study the projection quality, we use our error metrics presented in Chapter 4, applied to the two-dimensional projected points corresponding to a given viewpoint of the 3D projection. The best view (Fig. 5.12a) shows a very good projection quality in terms of both false neighbors (Fig. 5.12b) and also in terms of aggregated error (Fig. 5.12c). As we rotate the longest biplot axis towards its worst position (Fig. 5.12d, middle view), we see how both above-mentioned errors start increasing. In particular, we see high values for the aggregated error at the top of the projected point-cloud. When we reach the worst possible alignment for the selected biplot axis (Fig. 5.12g), where this axis is orthogonal to the screen, we clearly see a lot more false neighbors (Fig. 5.12h)) and many high values for the aggregated error. All in all, the experiment tells us that the alignment of biplot axes with the screen is a good measure of the projection quality. At a higher level, this tells us that our axis legends (Sec. 5.3.2), which quantify this alignment, are indeed good proxy representations for the quality of a projection.

In the second experiment, we want to study how the projection quality varies as a function of the alignment of the shortest biplot axis in a projection. For this, we use a similar dataset of four variables and 1000 observations. The first three variables are created identically to the first dataset. The fourth variable is generated by sampling an inverse function $f(x) = a/x$ with $a = 5000$ and $1 \leqslant x < 1000$, and corresponds to the shortest biplot axis. Figure 5.13 shows the vertical rotation of the short biplot axis from its position of best alignment with the screen to the worst alignment position where the axis is orthogonal to the screen. Looking at the corresponding error plots, we see that neither the false neighbors (Figs. 5.13b,e,h) nor the aggregated error (Figs. 5.13c,f,i) error values change much. In summary, by inspecting both our synthetic datasets, we can see a direct relation between the projection quality and the screen alignment of a biplot axis. If the axis is long, its alignment is critical to, and is capable of, producing a high-quality projection. If the axis is short, its alignment will not interfere much with the projection quality.

However, both above experiments involve artificial datasets. We still need to study how projection quality relates to biplot axis alignment for real-world datasets. To study this, we use the *wine* dataset which has 1599 points and 11 attributes. As for the previous two experiments, we use LAMP as projection method. The first observation to be made for this dataset is that it has has five long axes, two short axes, and three axes of intermediate length. This is a more complex situation than the earlier two synthetic cases, which included one long biplot axis, respectively one short one, from a total of four. We next notice that the longest biplot axis corresponds to the '2:citric_acid' variable. As for the first synthetic-dataset experiment, we study the error variation as function of the

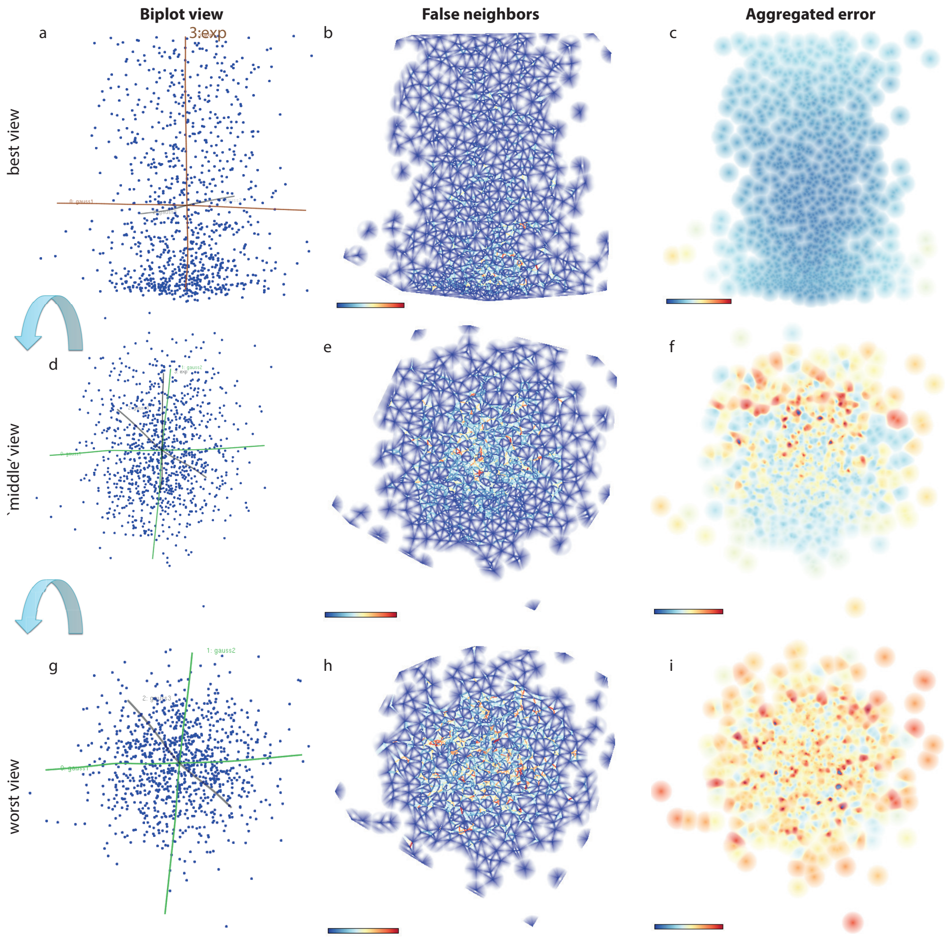


Figure 5.12: Vertical rotation of longest biplot axis (variable '3:exp') and corresponding aggregated error and false neighbors views, synthetic dataset.

alignment of this axis with the screen. Comparing the false-neighbor views for different alignments, we can see slight differences. For the best alignment viewpoint, false neighbors have more middle values (yellow edges), while the worst alignment shows higher values (red edges). Regarding the aggregated error, we see that the best alignment concentrates the middle and higher errors in the top-right of the point cloud, showing two outlier red points (highest error values). For the worst alignment, we see the appearance of one more red point, and also see that middle-range error values get more separated in two regions (top, down-left) in the projection. Repeating the experiment by varying the alignment of the shortest biplot axis (axis 3), we see a similar error variation as for the longest biplot axis alignment. For the worst alignment, the false-neighbors view shows more orange edges and slightly more red edges than the best alignment

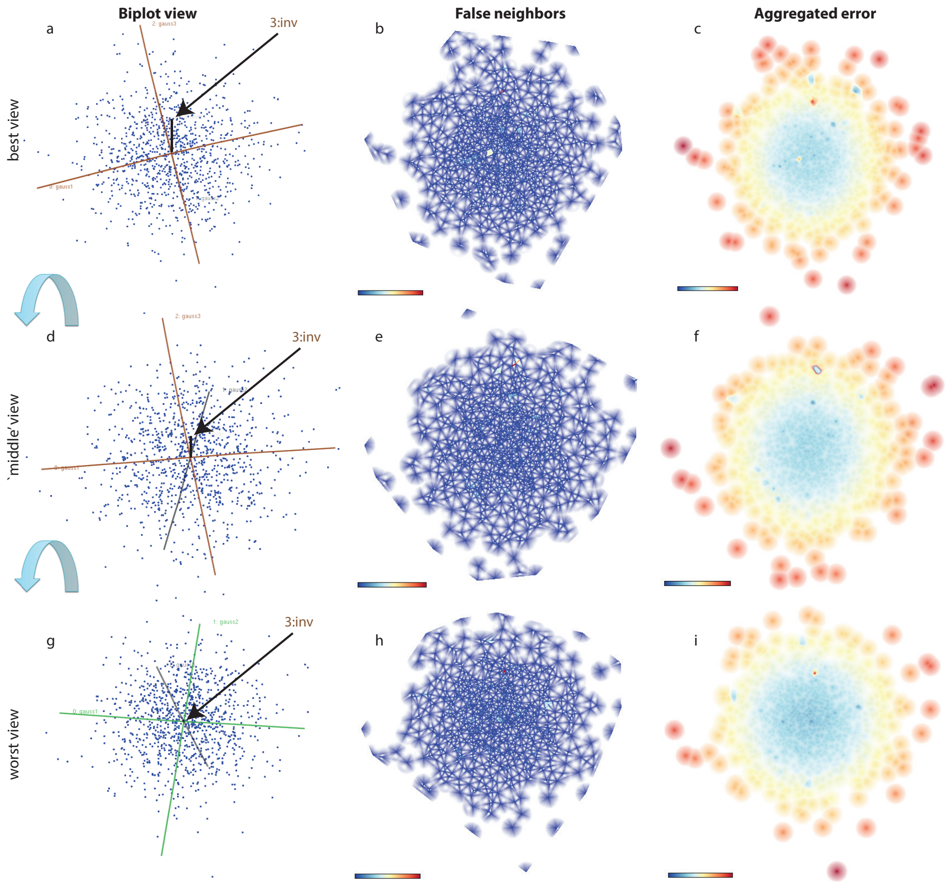


Figure 5.13: Vertical rotation of shortest biplot axis (variable '3:inv') and corresponding aggregated error and false neighbors views, synthetic dataset.

case. In the aggregated error view for the worst alignment, two additional red points appear as compared to the best alignment.

Summarizing the above observations, we can say that, for datasets which have a particularly long biplot axis, the alignment of this axis with respect to the screen does significantly influence the projection quality. Hence, the tools described earlier in this chapter for aligning axes to the screen have the additional value of being useful to control the projection quality, besides bringing specific data variations into focus. In contrast, for datasets which do not have such clear outlier biplot axes, the choice of viewpoint does not significantly affect the overall projection quality. Hence, the choice of viewpoint, for such datasets, should be based mainly on the interest of the viewer in examining particular variables, rather than on the projection quality.

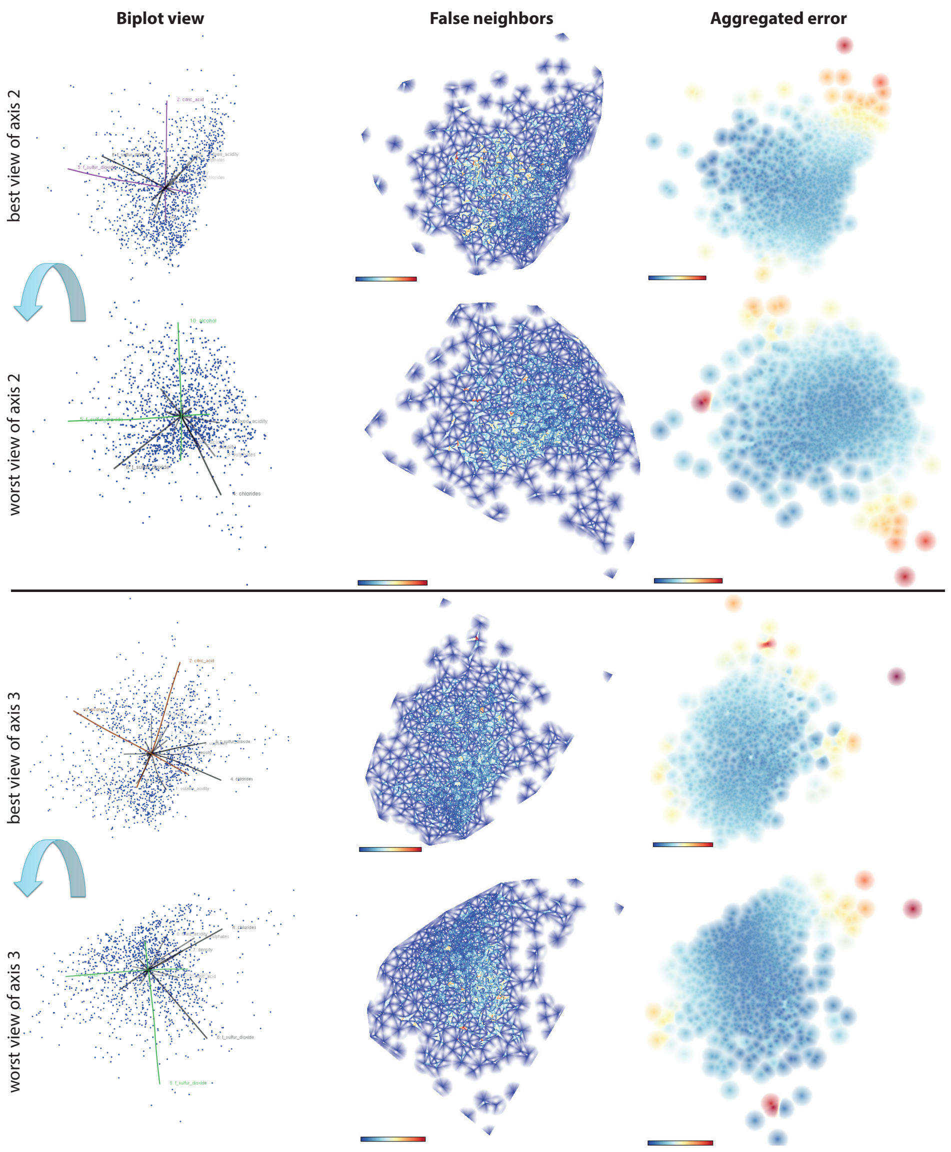


Figure 5.14: Best and worst viewpoints for the longest and shortest biplot axes, and corresponding false-neighbors and aggregated error views, real-world *wine* dataset.

5.5 DISCUSSION

Several points are relevant to discuss, as follows.

Scope: The effectiveness of our techniques depends, of course, on the quality of the DR projection and nature of the underlying nD dataset. If the projection captures distinct, well-separated, patterns in mD, our techniques will help explain the relationships of these patterns with the original n dimensions, and next choose good viewpoints to examine them. If the DR projection is suboptimally done, or if the input dataset does not exhibit any clearly segregated patterns, our techniques provide little additional insight in the data. So, our scope is to help users explain patterns, through in course correlations, the projected data in terms of the original variables, *if* such patterns exist in mD. If patterns are absent, one should use complementary techniques, outside the scope of our work, to improve the DR projection being used, *e.g.* [141]. Separately, if the nD data are clearly segregated into clusters *and* if one only wants to find such clusters, rather than the more fine-grained task of explaining spreads in the data or correlations of specific variables, our techniques are as useful as most state-of-the-art data clustering methods out there.

Generality: Our techniques work directly with any (non)linear DR technique that projects n variables to $m = 3$ dimensions, without needing to modify, or access the internals of, the DR technique. This is unlike [81, 1, 152, 28], which need to know that the DR being used is PCA or SVD to compute loading values. Our examples shown here use LAMP, ISOMAP, and FBDR as DR techniques. We have equally easily used LSP [163] and PLMP [164]. Other DR projection techniques can be equally easily used, with no changes to our proposal.

Scalability: Our methods are simple to implement and computationally scalable: We only need to apply the chosen DR projection to a small set of sample points distributed along the input variables (Sec. 5.3.1). For a dataset of D variables, N data points, and a number of n_{max} variables shown in the proposed legends, the complexities are $O(D)$ for the biplot calculation, $O(n_{max})$ for the axis legends, and $O(n_{max}^2)$ (for the viewpoint legend respectively. The memory complexity of the entire set of techniques is $O(N \cdot D)$, *i.e.*, equal to the size of the dataset to be stored. Visually, our axes legends, biplot axes, and viewpoint legend scale well up to roughly $n_{max} = 20$ variables, in line with other multivariate visualization techniques [152, 64, 28, 37]. When the input dataset has more variables, axis legends automatically show the n_{max} most visible variables for the current viewpoint, which is the best we can do in such situations (Sec. 5.3.2).

Comparison: Our axis-alignment and viewpoint legends have some similarities (and differences) with 'rolling the dice' (RTD) [64]. Our axis-alignment (Sec. 5.3.1) and best-viewpoint tools (Sec. 5.3.3) resemble the scatterplot-matrix cells in the sense of selecting 'interesting' variable-pairs. Yet, while RTD defines these configurations as variable pairs mapped to Cartesian scatterplots, we define these as viewpoints in a 3D space *given* by the DR projection that can best highlight variable-combinations of interest. Since we cannot control the DR projection, our viewpoints can show orthogonal biplot axes, but also slanted and/or curved axes of different lengths. Also, our viewpoints show, by construction, *all* projected axes, rather than a fixed subset of two. Finding a good data-exploration sequence is equivalent, in our case, to finding a navigation-path between highlights on the viewpoint legend sphere. The main added value of the viewpoint legend is that it shows *all* possible viewpoints in-between these highlights.

Technical details: Our categorical, continuous colormaps, and transfer function choices (Sec. 5.3.3) are, of course, open. Better alternatives may exist for specific user groups and work domains. We used simple and well-known presets for these designs precisely to make it easier to separate our contributions from such specific design elements.

Evaluation: We evaluated the proposed techniques on 9 datasets (300 to 200K points, and 6 to 25 variables). Learning to interpret the axes biplots, axes legends, and viewpoint legend was perceived as very simple and intuitive, mainly due to the fact that all these visualizations are interactive and dynamically change as the user rotates the viewpoint. Besides the selection of the variable used to color points, our techniques do not require any explicit parameter user-setting. Compared to classical 2D scatterplots, our techniques need additional time to learn them (around 20 minutes, as observed by explaining them to 9 users not involved in this work) – which is in line with learning times reported in [64, 28] for similar tasks and user counts. Users found the biplot axes easiest to understand and use, arguably due to the fact that similar axes appear in many types of plots. The interactive axis alignment described in Sec. 3.2 was also found simple to understand and use, as it requires basically two clicks in the desired bars of the x and y axis legends. Using the viewpoint legend was perceived as the most complicated, as this widget requires memorizing the appearance of several large same-color areas on the surface of the sphere while interactively rotating the viewpoint. We acknowledge that these findings need more refinement and validation, *e.g.* in terms of a controlled user study.

Limitations: Large 3D DR scatterplots inherently generate occlusion which, even with transparency and interaction, can be hard to disambiguate. Biplot axes for a few highly non-linear projections (*e.g.* force-based methods [214, 171]) are highly curved. Yet, such methods are not preferred, precisely because of their error rates and the difficulty of finding globally good viewpoints, and thus affect our overall proposal only marginally. Our tools do not aim to fully remove interactive trial-and-error exploration, such as brushing or viewpoint selection. Their added-value is to make interaction more *targeted* towards a given goal – *e.g.*, when (slightly) changing a viewpoint, one immediately sees the effect on the axis biplots, axis legends, and viewpoint legend, and thus can better estimate what to expect to see when turning the viewpoint this or that way; when one wants to examine one or two specific variables in context, we allow doing this by just two clicks on the axis-legend bars for those variables. Separately, we note that our examples in Sec. 5.4 do not imply that 3D projections are always bet for addressing all related tasks: rather, we show how 3D DR projections, *if* chosen for the sake of minimizing distance errors, can be made more effective as compared to raw 3D scatterplots.

5.6 conclusion

We have presented a set of interactive visualizations that help users explore and explain 3D dimensionality-reduction (DR) projections of high-dimensional data. Our methods, realized as linked views, explain the meaning of projected dimensions in terms of original variables; show projection nonlinearities and correlations (or lack thereof) for these variables; help finding good viewpoints from which given variable-pairs can be best explored; and quickly show which variable-pairs can be explored from any possible viewpoint. Globally, our techniques aim to help users interpret raw 3D projections in typical xy scatterplot terms. Our techniques are easy to implement, scale well computationally and visually, and can be added in a non-intrusive way to any DR technique as extra aids to classical brushing and color-mapping explanatory tools.

Separately, we analyzed how the quality of 2D views of such 3D projections varies in function of the biplot axis alignment with the screen, by using the projection error views introduced in Chapter 4. The performed experiments show that, for datasets exhibiting long biplot axes, their screen alignment does matter for projection quality. This gives an additional value to the axis and viewpoint legends presented in this chapter, which can thus be used to estimate, and improve, the projection error corresponding to a given axis alignment.

At a high level, the tools presented here, and corresponding application examples, support the view that 3D dimensionality-reduction projections *can* be easy to explore in practice. Given that their overall projection error is known to be lower than for corresponding 2D projections, this makes 3D projections an interesting practical alternative to the use of the better-known 2D projections. We shall explore the comparative added value of 2D projections and 3D projections using our interactive explanatory tools further in Chapter 6 in the context of a concrete application.

Future work targets enhancing the insight given by our explanatory visualizations, by studying how the local nonlinearity of projections, and local projection errors, can be better and more intuitively conveyed for 3D projections of large datasets. Validating the value of our visualizations via user studies is a second important future work topic.

6 THE SHAPE OF THE GAME

ABSTRACT: *Sports videos, such as soccer matches, are a prime example of rich multimedia content of high interest to a broad public. This has been recognized by many researchers who focus on analytic solutions for such data, such as the online detection of interesting events, analysis of team strategies, or comparison of teams' statistics. Significant research also focuses on efficient algorithms to detect match statistics, strategy, retrieval and indexing or summarization. In contrast, the problem of presenting such information to the casual end user is studied much less. In this chapter, we address this last goal, by proposing a simple but efficient visual metaphor to help non-specialist users browse and get insight in soccer matches. For this, we extract video segments, based on audio and metadata, identifying the main events according to the narrator's emotion. We next use such events to create a visual representation that preserves the video sequence but highlights the most important events. The proposed visualization enables the quick identification and navigation to the main events of a soccer video, and also a way to compare different matches and entire tournaments. To understand the added value of our visualization, we perform several user studies involving our proposed video-browsing tool, and also compare it with the exploration of the same data by using classical multidimensional projections.*

6.1 INTRODUCTION

The development of multimedia and network technologies provides an increasing presence of video content over broadcasting and streaming services. Recently, the 2014 World Cup soccer tournament registered record-breaking audiences all over the world [100], becoming the most accessible edition of a soccer tournament in history, reaching up to 5.9 billion screens all over the world [155]. In line with this phenomenon, there is a steady increase of interest in soccer video analysis [59] involving multimedia information retrieval, video indexing and processing, video semantic analysis, and video visualization. Soccer videos are a rich form of multimedia content, consisting of several audio and/or video tracks, subtitles, and related metadata. As such, they form a interesting target for our exploration of the possibilities of visual analysis of multidimensional multimedia data.

Different kinds of insights and information can be identified using soccer video analysis [154]. Users who wish to see only important or interesting events, also known as *highlights*, are best served by match summarization systems [213, 185, 150]. Users who want to see the team strategy, by *e.g.* tracking the ball and/or players in the field, are best served by tactical-information systems [137, 235, 172]. Systems that compute match statistics are useful when one wants a quantitative game analysis including red and yellow cards, ball possession, shots on target, and goals [188, 115, 173]. Finally, users who want to find similar matches or match fragments are best served by search-by-example systems using video retrieval and indexing techniques [210, 118].

There are two fundamentally different approaches to getting insight into such data, as follows.

Data-centric approach: The first such approach is a data-centric one. In this approach, we consider out multimedia data as a multidimensional dataset, where observations encode all available data measured at different time instants, and dimensions encode the different attributes being measured at such time instants, respectively. Next, we can use the various visualization techniques designed for multidimensional data to explore this dataset. Since the focus of this thesis is on multidimensional projections, natural questions that arise is are how effective are projections for this exploration, and what type of insights can be obtained from our multimedia data by using them?

User-centric approach: While the above data-centric approach has the potential to reveal interesting insights, it is arguably not the most effective one for certain end-users interested in our soccer multimedia data. Multidimensional projections are not particularly easy to use, or appealing for, casual users such as typical soccer fans. For instance, such visual metaphors may not be the optimal ones to help our casual users to easily answer questions like "Which events are interesting in this match?", "Are there any polemic or controversial events?", "Which team has the best performance?", "When do the main events occur in the match?". The same problems occur, even more prominently, for tasks related to comparing different matches or analyzing entire tournaments. With such users in mind, and their questions, other visualization designs may be better.

Summarizing the above, we identify two related research questions:

1. How much, and what types of, insight can we get from analyzing only the audio and metadata dimensions of a soccer multimedia dataset, using multidimensional projections?

2. Are other visualization designs, apart from multidimensional projections, able to convey different insights, and in an easier way, to casual end users such as sports fans?

The two above questions are inter-related, since they both aim to explore the same type of data (by different methods), and we are interested to identify the advantages and limitations of these different methods. However, they also have independent components, since they target different users with potentially different types of questions. To answer the above questions, we propose a two-step approach, as follows.

In the first step, we study our multimedia data from a *user-centric* exploration perspective. For this, we first extract a number of relevant attributes from the multimedia data. In detail, we extract video segments, which we further call "video skims", based on audio and metadata, so that the main events of the match are identified according to the narrator's emotion. Next, we use such skims to create a visual representation that preserves the video's temporal sequence but also highlights the most important events. The proposed visual representation enables an easy identification of the events of interest and allows users to quickly navigate to, or between, them when exploring a match. Additionally, our visualization supports exploration of matches at coarser scales, such as evaluating the quality of a match or parts thereof and comparing different matches or tournaments. Interacting with the proposed visual representation is very simple, and can be done by using a typical remote controller, a metaphor which is arguably well known by our end-user group.

In the second step, we take a *data-centric* approach. We regard the time-dependent events extracted from the soccer multimedia data as a multidimensional dataset. Next, we use classical two-dimensional and three-dimensional projections, and their associated interactive exploration tools, to explore this dataset, searching for various types of insights. This approach offers a fundamentally different way to look at the (same) underlying data as compared to the first approach outlined above. In particular, events of a more fine-grained nature can be located, and global correlations between the measured attributes can be studied. By comparing the types of insights obtained with the two visualization approaches, and the interaction and reasoning patterns involved in obtaining these insights, we draw several conclusions regarding the suitability of multidimensional projections as tools for exploring multidimensional data obtained from multimedia content, as a function of the types of questions being posed and the types of users involved in the exploration.

In Section 6.2, we overview related work in sport visualization and match summarization. Section 6.3 presents our first approach, the user-centric approach to exploration of sports videos aimed at sports fans. We present this approach first, since it is simpler to follow than our second multidimensional data-centric approach, and it also offers a quick introduction to the structure of our multimedia data. Section 6.4 presents experimental results of our user-centric visualization approach. Section 6.5 presents our second approach for exploring sports videos using multidimensional projections, aimed at more specialized users. Section 6.6 discusses and compares our two visualization approaches, and thereby outlines advantages and limitations of the multidimensional projection exploration metaphor. Section 6.7 concludes the chapter.

6.2 RELATED WORK

Due to the huge popularity of soccer, several solutions have been proposed to extract insights from soccer videos. At a global level, we can divide these solutions into two categories, as follows.

Specialized systems: These solutions typically present complex information extracted from the sports multimedia data at a relatively high level of detail, and are aimed for professionals such as trainers, sports analysts, and sports commentators. From a technical perspective, these solutions are designed similar to many other visual analytics systems, offering multiple linked views for exploring the extracted events from the multimedia data, detecting patterns and outliers, and creating and iteratively refining hypotheses on the underlying phenomena. From a visualization perspective, the main contributions in this class target the presentation of statistical [188, 115, 173] and tactical [137, 235, 172] soccer match information. Improving upon the traditional soccer ranking tables, À Table! provides temporal navigation in two different views [173]: dynamic animation over the rows of the ranking table and a transient line-chart of team ranks to visually explore the performance of teams in a championship. Khacharem et al. use animation to convey dynamic time-dependent data such as motion and trajectory in a single display [115]. They also observed that non-expert users preferred the simpler static visualizations, while experts preferred the more complex dynamic ones. Soccer Scoop proposes two visualizations (field and player viewer) to show and compare the dynamics of distinct players, for the user group consisting of team managers [188]. Based on the visualized statistics, team managers were able to determine strategic insights, like if a particular player plays better on the road or at home. Similar insights and use-cases were presented by Lucey et al. who analyze spatiotemporal data obtained by tracking the ball in English Premier League matches [137]. In particular, they showed how hypotheses such as "win at home and draw away" can be checked based purely on the video data. Spatiotemporal tracking of player positions was used in [235] to detect team-formation patterns associated to match events. SoccerStories conveys tactical and statistical information in a single view, showing player actions and ball shots at the same time of game and individual player statistics [172].

End-user systems: Apart from the above more advanced visualization systems, which aim to provide tactical and statistical information to expert users, several other systems have been designed for the casual user (typical soccer fan), such as video summarization [213, 185, 150] and content retrieval systems [210, 118]. In this context, the main aims are to help casual users to save time while browsing match videos, by attracting attention to the main match events. The main goal of video summarization is to detect typical events of interest (*e.g.* foul, goal, shot, corner, offside) [213], or detect the main match events, also known as highlights [185, 150]. For example, the video summarization in [213] uses Bayesian networks to classify and detect, through "play-break" semantic units, seven different event types: goal, card, goal attempt, corner, offside, foul and non-highlight. This idea was extended to capture not only game highlights, but also competition-intensive scenes and emotional events, which are presented as frame sequences [150]. Soccer highlights can be automatically generated from audio and video descriptors [185]. After segmenting the video into shots, audio detectors including an algorithm for referee's whistle were used to create the highlights. For retrieval systems, CrowdSport uses crowd-related information to generate semantic annotations and create short video snippets enriched with event information such as team behavior and individual player motion [210]. Relatedly, Kolekar *et al.* used a probabilistic Bayesian belief network (BBN) method to index important soccer and cricket video fragments detected by semantic concept-labels and audio features [118]. This work could also be considered as video summarization, since they provide to the end user a set of video clips for each specific event detected.

Summarizing all the above, we see that, while many efficient and effective methods for video summarization and event extraction have been proposed, less attention has been dedicated to *present* the summarized or indexed information in an easy-to-understand and easy-to-use way to the casual watcher. As such, our first visualization approach, described next, focuses on presenting a soccer video summarization approach based on a simple and easy-to-understand visual metaphor.

6.3 USER-CENTRIC APPROACH: VIDEOPLAYER-STYLE SPORTS VIDEO SUMARIZATION

To tackle the problem of summarizing sport videos and providing a representation to speed-up video browsing, we propose a two-step approach (see also Fig. 6.1). In the first step, the most important or interesting events, the highlights, are identified based on (a) sound information extracted from the actual video and (b) metadata retrieved from the Internet that represents various match statistics, such as goals, cards (yellow/red) and substitutions. In the second step, the video is segmented, based on the detected highlights, into multiple small segments, the 'video skims', that are used to create the visual representation of the match. Users can interact with this representation to focus on the most interesting moments, get more insight on specific moments or match phases, and compare several matches. These steps are detailed next.

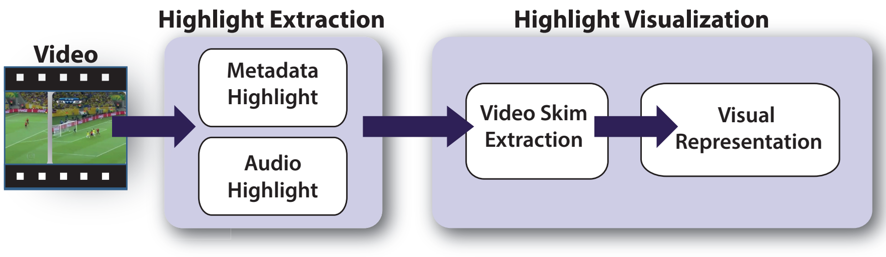


Figure 6.1: Pipeline of the construction of our visual summarization.

6.3.1 *Detection of audio highlights*

The sound track of the major broadcasted sports – soccer in particular – is composed mainly of foreground commentary coexisting with background noise. Essential to match summarization, the most important events can be identified by analyzing both the foreground sound (which contains the narrator's speech) and also the background sound (which contains audience cheering and applause and/or the referee whistle). A common feature to describe highlights is the sound volume or loudness [126].

In our approach, similarly to [126], we detect highlights based on the audio loudness. For this, we first divide the audio signal into short sub-segments, called *clips* [232]. A clip consists of a certain number of partially overlapping audio-frames, depending on the sampling frequency. In our experiments, each clip is a one second time-interval in a mono channel, sampled at 44.1kHz, resulting in 86 audio-frames, each frame composed by 1024 consecutive audio samples with 50% overlap between adjacent frames (512 samples in common with the previous frame). The loudness $l_{frame}(k)$ of a frame k is given by the RMS (Root Mean Square) of the audio signal magnitude, *i.e.*

$$l_{frame}(k) = \sqrt{\frac{1}{N} \sum_{n=0}^{N-1} y_{k,n}^2},$$

where $\{y_{k,n}\}_{0 \leqslant n < N}$ is the set of $N = 1024$ audio samples of frame k. For each clip c, we next compute its audio loudness $l_{clip}(c)$ as the mean value of all frame loudnesses in the clip, *i.e.*

$$l_{clip}(c) = \frac{1}{86} \sum_{k \in c} l_{frame}(k), \tag{6.1}$$

Based on this information, we next define highlights as those clips with the largest loudness in the video that are not neighbors, considering a window of size $2t$, of a clip with a larger loudness. We set $t = 7$ seconds as we do not expect that consecutive highlights occur in less than 7 seconds in a soccer match. Separately, the number of identified highlights is limited by the user according to the display area available, or considering a minimum value of loudness (as discussed further in Sec. 6.4).

6.3.2 *Detection of metadata highlights*

Besides extracting highlights by analyzing the video content, one can use alternative information sources, such as external text sources. Also known as metadata, this type of data can enhance semantic analysis since it can provide valuable information such as descriptions and time information of match events. In our work, we used metadata from an open-source soccer API describing all 2014 World Cup matches (see Sec. 6.4.1). This API provides descriptions about three different types of events (goals, substitutions and cards) and their respective time stamps. Goal events also include the scoring player's name. Substitution events mention the entering and exiting player names. Card events indicate the card type (color) and involved player. One highlight is extracted per event.

We next show how the metadata+sound highlights are used both for segmenting the video and for creating the visual representations of important events during a match.

6.3.3 *Constructing the visual representation*

To show the highlights, we chose to use a video-abstraction artifact called 'video skim' [221] rather than traditional keyframes, due to the higher summarization power that such video skims offer as opposed to static images. In our approach, a video skim is a video segment centered on a detected highlight, having a duration of $2t = 15$ seconds (Sec. 6.3.1). Making the length of a video skim equal to $2t$ guarantees that no important highlight is lost and that each highlight is represented by a single video skim. Extracted video skims are resized to reflect the *importance* of the respective highlights they represent. Below, we first discuss how the importance is computed, based on the audio signal and/or the video metadata. Next, we discuss how video skims are resized and assembled to produce the final visualization.

Audio highlights importance: The importance $s^A(v_i)$ of a video skim v_i extracted considering the audio highlight corresponding to the clip c_i (Sec. 6.3.1) is computed based on the loudness of c_i as

$$s^A(v_i) = \left[\left(\frac{l_{clip}(c_i) - \rho_{min}}{\rho_{max} - \rho_{min}}\right) + 1\right]^2 \tag{6.2}$$

where $l_{clip}(c_i)$ is the loudness (Eqn. 6.1) of the clip c_i, and ρ_{max} and ρ_{min} are the maximum and minimum loudness values of the entire video, respectively. The importance $s^A(v_i)$ ranges between 1 and 4, with larger values indicating more important audio events.

Metadata highlights importance: As with the audio highlights, an importance $s^M(v_i)$ is calculated for each video skim v_i that is extracted from the metadata highlights. In contrast to audio highlights (Eqn. 6.2), the importance of metadata highlights is computed as

$$s^M(v_i) = \begin{cases} 1.0 & \text{if} \quad \mathit{type}(v_i) = \mathit{substitution} \\ 2.0 & \text{if} \quad \mathit{type}(v_i) = \mathit{yellowcard} \\ 3.0 & \text{if} \quad \mathit{type}(v_i) = \mathit{redcard} \\ 4.0 & \text{if} \quad \mathit{type}(v_i) = \mathit{goal} \end{cases} \tag{6.3}$$

where $\mathit{type}(v_i)$ gives the type of metadata event v_i. The importance values in Eqn. 6.3 were chosen empirically, favoring goals as the most important and substitutions as least important. Note also that both s^A and s^M take values in the same range $[1, 4]$.

Combining overlapping video skims: To compose the final visualization, the video skims corresponding to audio and metadata events are combined. This merges thus salient event information (such as goals or cards) with information that captures the narrator's excitement and polemic/controversial or even uncommon events (as captured by audio highlights), thereby presenting to the casual user a more complete match overview. If two video skims v_i and v_j, corresponding to events extracted from the audio, respectively metadata, are found to overlap in time, we merge them into a single skim v whose importance $s(v) = (s^A(v_i) + s^M(v_j))/2$ is set to the average of the importances of the merged skims, and whose time extent $v = v_i \cup v_j$ corresponds to the union of the two skims.

Visual layout: After the generation of the combined audio-and-metadata video-skim stream $V = \{v_i\}$, we resize each skim v_i in V to reflect its importance $s(v_i)$. The underlying idea is to make skims larger for more important events, so that users see these saliently in the resulting visualization. Additionally, we limit the size of the smallest video skim (corresponding to $\min_{v \in V} s(v)$) to 50 pixels, to ensure its visibility, and the size of the largest video skim (corresponding corresponding to $\max_{v \in V} s(v)$ to four times the minimal size, to limit the utilized screen space.

After skims are resized, we organize them sequentially along the x screen axis, which encodes thus the video timeline. In detail, skims are vertically centered along this timeline, and horizontally aligned with respect to their temporal order, so that consecutive skims v_i and v_{i+1} have a small overlap of a few pixels. We use this overlap to further emphasize the skims' importance. For this, skims are rendered back-to-front in order of increasing importance, so that more important skims slightly overlap less important adjacent skims. By adding a faint shadow around the borders of each skim, this pseudo-depth effect further emphasizes important skims (large, overlapping, and appearing in front)

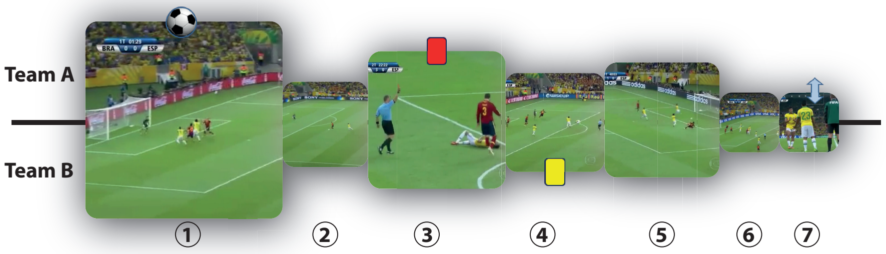


Figure 6.2: Example of the visual outcome created by our technique. The skims with icons (1,3,4,7) represent events detected using the metadata. The other skims (2,5,6) were detected using only the audio data.

as opposed to less important skims (small, overlapped, and appearing in the back). Finally, we also map skim importance to the saturation of the respective video skims, thereby making important events more colorful and less important ones gray. Overall, the above four cues (size, overlap, shadows, and saturation) jointly help observers in quickly finding the most important skims, and also locally sorting skims by relative importance.

The three available types of information provided by metadata (goals, substitutions and cards) are mapped as small icons over the corresponding video skim, showing if it represents a goal, substitution or yellow/red card respectively. If the icon is added to the top of the video skim, it refers to an event related to *Team A*, if it appears on the bottom it refers to event related to *Team B*. Figure 6.2 shows a synthetic example of a visual summarization created by our technique. Here, event **(1)** represents a goal for *Team A*. Events **(3)** and **(4)** represent a red card for *Team A* and a yellow card for *Team B*, respectively. Event **(7)** is a substitution for *Team A*. Events **(2)**, **(5)**, and **(6)** were detected using the audio data only, and represent other types of events.

We use the above skim layout to construct two separate timelines, one for each of the two halves of a soccer match. For matches having additional extra-time, we add a third timeline below the two main ones.

6.4 user-centric approach: results and evaluation

6.4.1 *Datasets*

Finding publicly available sports datasets which include soccer metadata is not an easy task. To overcome the locked nature restrictions of today's proprietary sports datasets, Bergmann et al. [17] presented soccer data (matches, teams and players information) as a Linked Open Data solution (LOD) collected from heterogeneous sources and linked to LOD datasets. Some alternatives to open soccer data exist, such as openLigaDB[1] and the Openfooty API[2]. Both websites contain soccer data statistics. The openLigaDB site makes sports data for public use via web services; however, it is restricted to German language only. In contrast, the Openfooty API has media content but does not describe recipes in detail and is subject to restrictions that forbid republication as Linked Data. Given the above limitations, we based our work on matches from the World Cup 2014 and UEFA Champions League 2015. Metadata was collected manually from the FIFA official website[3]. Corresponding video data was obtained separately from different Internet sources, and includes matches recorded at different resolutions and produced by different broadcasters.

We next describe the analysis process performed on these data, starting with the video stream analysis and ending with the usage of our summary visualization.

6.4.2 *Visual exploration*

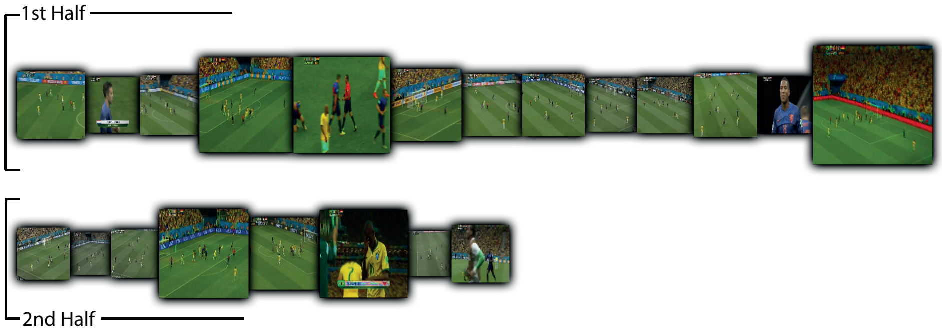


Figure 6.3: Visualization constructed from audio highlights only. In this representation the match is summarized according to the narrator's emotion and audience excitement.

Audio-based exploration: As a first element, our visualization allows users to interpret the match statistics and the narrator's emotion and audience excitement. When constructed only based on highlights extracted from audio data, the visual representation will capture the emotion and excitement as a function of the sound loudness. Figure 6.3 presents the visual outcome for the Brazil-Netherlands match for the third place in the 2014 World Cup considering audio highlights only. The audio is in the English language. The presented visualization allows one to temporally navigate through the match highlights, and easily get the most excitement moments of a match. In the first half of the match three

1 http://www.openligadb.de/
2 http://www.footytube.com/openfooty/
3 http://www.fifa.com

main events are detected, two refer to goals of Netherlands and one is a clear chance of goal for the Brazilian team. In the second half, two main events are detected, a yellow card for Brazil, resulting from a penalty simulation, and another goal for Netherlands.

To provide more insight, and support answering additional questions, we provide user interaction, as follows. A mouse-over movement on a video skim sets the image used to depict the video skim to the time moment corresponding to the mouse x position. This way, moving the mouse to the left or right over a skim basically plays the video skim forward or backward respectively. A left-button click on the timeline plays the underlying video skim. Finally, if the user wants to see more detail, a right-button click pops the underlying video skim in the screen center and plays it. When playback is ready, the skim pops back to its original space in the visualization.

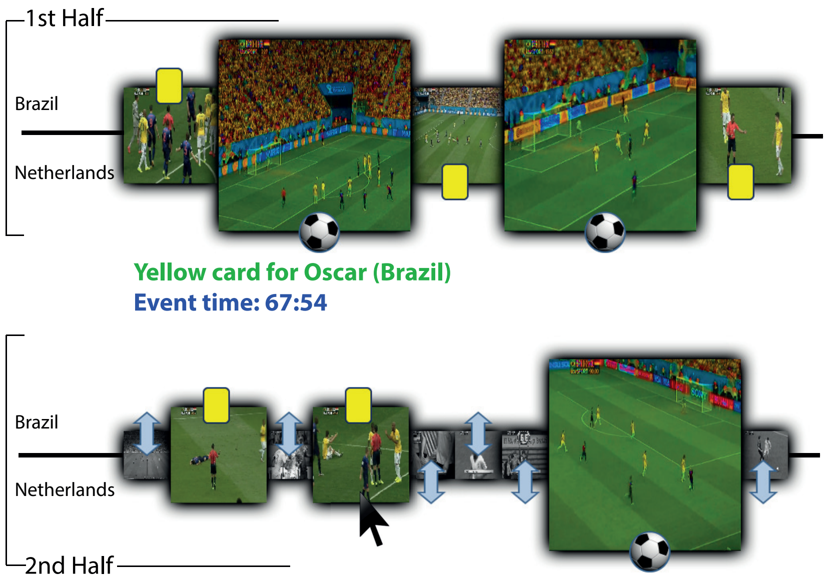


Figure 6.4: Visualization constructed from metadata information only.

Metadata-based exploration: In contrast to the above, visualizations constructed from metadata only support scenarios where one wants to focus on statistical information, *e.g.* answer questions like: "Which events represent goals, cards, substitutions, and to which team do these belongs?". Figure 6.4 shows such a metadata-only visualization of the same match as in Figure 6.3. As explained in Sec. 6.3.3, icons indicate metadata events: a soccer ball shows a goal; yellow and red rectangles show yellow and red cards, respectively; and colored arrows show player substitutions. The annotation locations (top or bottom of the skims) indicates if the event relates to the home team or away team respectively.

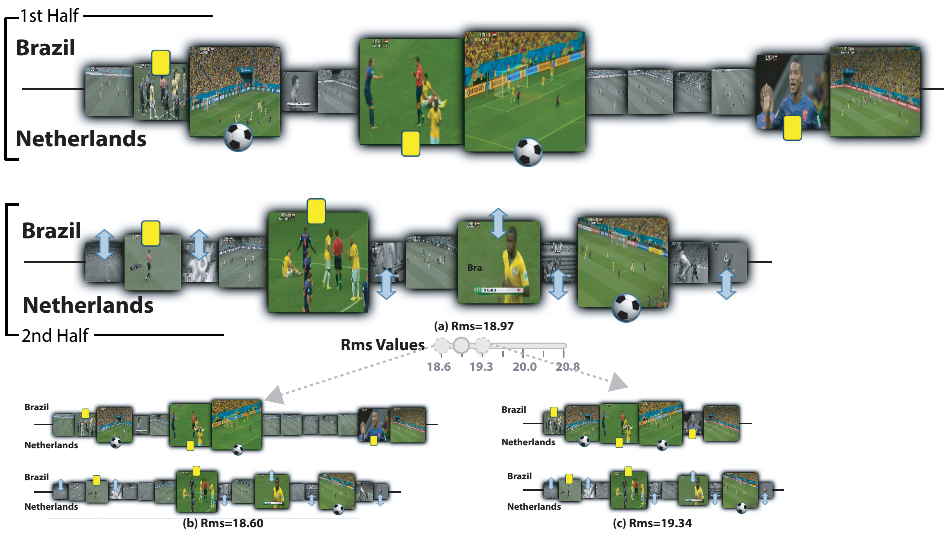


Figure 6.5: Visualization using audio and metadata information with different levels of detail. The most important events are captured, some only by the metadata (substitutions) and some only by the audio (the goal chances).

In contrast to the audio-based visualization, frames shown in the video skims correspond now to the precise moments recorded by the respective events – *e.g.* the two first-half goals, complaining moment of players after the first yellow card, and lastly the three Dutch substitutions in the second-half, where the camera zooms on the player going out (Fig. 6.4). In contrast to the audio-based skims, mouse-over interaction for metadata-based skims shows additional information about the event as text annotations – *e.g.* the information about the yellow card received by Oscar (mouse over the second video skim, second half, Fig. 6.4).

Combined audio+metadata exploration: Apart from audio-based and metadata-based visualizations, we can combine both types of highlights to convey more information and/or provide more accurate answers to questions such as "Which are the most interesting events?" Figure 6.5 illustrates this multimodal (audio+metadata) approach combining the highlights of Figures 6.3 and 6.4. The visualization's level-of-detail is controlled by a slider, filtering the audio highlights according to a minimum value of loudness (Eqn. 6.1). Moving this slider thus controls how many audio highlights are shown. All metadata-based skims are, however, kept and multiplexed with the audio-based skims, since metadata are typically of high accuracy and importance [154].

Figure 6.5a shows a situation where the user selected a minimum loudness of 18.9; the resulting visualization shows all metadata highlights and audio highlights louder than the given threshold – 25 highlights in total for the entire match. If the loudness slider is decreased to 18.6, 30 highlights are selected (Fig. 6.5b).

Conversely, increasing the minimal loudness to 19.3 shows only 20 highlights (Fig. 6.5c)). Note how, irrespective of the selected level-of-detail, all goal events are kept *and* mapped as large (thus, important) video skims.

A separate unexpected finding is the important role of the audience's cheering in the background: In Fig. 6.5a-c, the rightmost large video skim in the first half denotes the best chance Brazil had to score in the match. Brazilian audience was by far the largest in the stadium, as they were supporting their home team. This explains why some goal video-skims are not as large as expected. Separately, both yellow cards for Netherlands were strongly celebrated by the audience in the first half of the match, causing larger video skims than what a metadata-only visualization would have made. The same can be observed for the yellow cards on the second half of the match. The second yellow card has a larger importance than the first one since it results from a penalty simulation and its importance is increase due to the audience's cheering. Finally, the use of audio highlighted some shots on goal by Brazil, and faults made by Netherlands, which are not present in the metadata information that records only goals, cards, and substitutions. Overall, the above points highlight the added-value of the combined audio+metadata visualization as opposed to audio-only or metadata-only approaches.

Large-scale analysis: For further insight, we used our proposed combined audio-and-metadata visualization to analyze and understand a portion of the last world-scale soccer event (2014 World Cup). From this, we consider here the set of matches played by the finalists (Germany and Argentina) in the so-called knockout stage. Tournaments usually adopt this stage since the traditional point-based system is too long; all teams play against each other; and also because this creates more exciting dynamics, as the losing team is out. It is exactly this excitement that we want to recover using our proposed visualization.

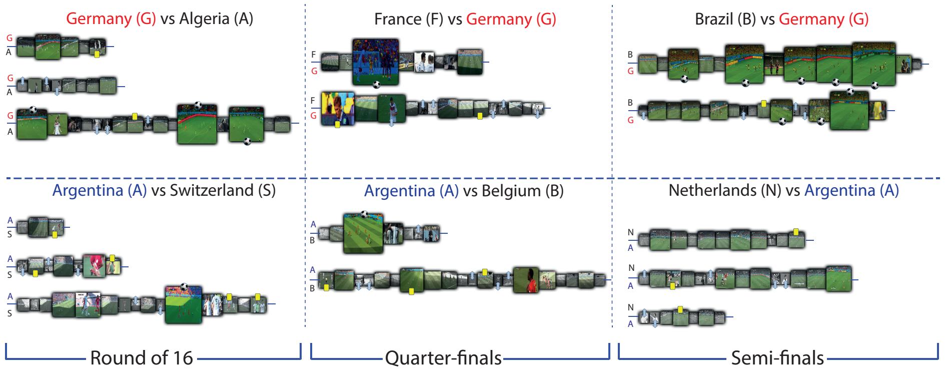

Germany (G) vs Algeria (A)
France (F) vs Germany (G)
Brazil (B) vs Germany (G)
Argentina (A) vs Switzerland (S)
Argentina (A) vs Belgium (B)
Netherlands (N) vs Argentina (A)
Round of 16
Quarter-finals
Semi-finals

Figure 6.6: Dynamics of finalists in the knockout stage of the 2014 World Cup.

Figure 6.6 illustrates the finalists' dynamics in the knockout stage, excluding the final match (discussed separately below). One rapid first insight is visible in the top-right: Argentina had to face two extra-time matches (*vs* Switzerland and Netherlands) before the final, while Germany only one (*vs* Algeria). Both matches of round of 16 can be classified as "boring" matches during the regular time, but very exciting on the extra-times, most probable because all goals were scored in the respective extra-times, 3 goals in Germany *vs* Algeria (score: 2 *vs* 1) and one in Argentina *vs* Switzerland (score: 1 *vs* 0). Quarter-finals were quite balanced, both teams scored only one goal and both at the beginning of the first half, (largest-third video skim, first-half row, Fig. 6.6), which are the largest skims in both matches. Last but not least, the semi-finals were marked by the largest tournament score in the Brazil *vs* Germany game, easily visible in the respective visualization by the many large skims, present mainly in the first-half when Germany scored 5 out of its 7 goals. Interesting to observe is that the importance of these 5 goals increase during the time, probably indicating that the audience's hope (predominantly composed by Brazilians), given by the small cheering on the first goal, turns into complaints on the last one. In the second half and second half-time, Germany scored twice and Brazil got her unique goal. A quite different scenario is visible in the Netherlands *vs* Argentina match, where there were no goals during regular time, but Argentina won in the penalty shootouts. Although such match do not present goals on the regular or extra times, this was a very exciting match, reflected on the number of highlights captured by the sound.

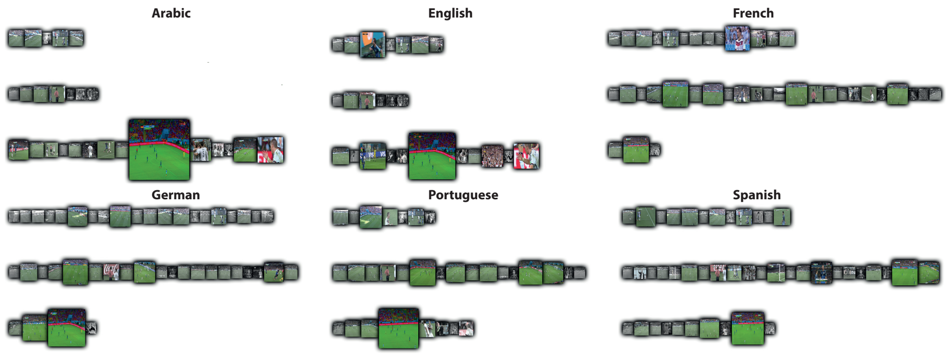


Figure 6.7: Visualization of dynamics of 2014 World Cup final match in six different languages. Different cultures define different views of a match. Our visual metaphor can be used to easily verify and analyze cultural aspects.

As we expect that our method captures the most exciting (and/or important) audio events, it is separately interesting to analyze the specificity of the sound track. Clearly, audio variations are expected between different broadcasters, as each one has different broadcast styles, commentator voices, and audio post-processing (*e.g.*, smoothing out noise). Still, one interesting question is: Which variations can be visibly measured considering not only different broadcasters, but also different languages? Does the same match come over differently?

To get insight in the above, we looked at the 2014 World Cup final match in six different languages/channels (Arabic/Bein; English/BBC; French/TF1; German/Das Erste; Portuguese/Globo; and Spanish/Gol), see Fig. 6.7. From a quick look, it is obvious that the displayed visualizations are very different and do not show the same dynamics. For example, the number of important highlights (large skims) is quite different; also, given the same loudness threshold, visualizations show quite different numbers of highlights, meaning that the excitement level was quite different. For example, if we look at the less salient (thus, having smaller skims) half-time, we see that Arabic and Portuguese broadcasters were less excited than other broadcasters; in contrast, the English had a more muted second-half, and the French show less highlights in extra-time – notice that the French narrator give no importance to the German goal in the extra-time (why?). Excluding Arabic and English, the remaining broadcasts show a very excited second-half with many highlights, in special French and Portuguese. For the extra-time, Arabic and English prevailed with larger and many highlights. By clicking on the most-salient skims in the respective visualizations, and listening to the respective videos, we find another interesting fact: When a goal is scored, the Arabic, Portuguese and Spanish yelled "Gooaal" in their respective languages, while the rest chose to call the scoring team or player name. Separately, Argentina had a disallowed goal event in the first half; the English broadcast was the only one to capture this event. This is an example of our tool being able to support answering questions such as: "Were there some polemic or controversial events?"

6.4.3 *User Evaluation*

We have performed a preliminary user evaluation to assess the usability of our proposed visualization. For this test, we recruited 7 participants. As a first step, all participants watched together the match Barcelona *vs* Manchester City for the *UEFA Champions League*, the second match of the round of 16, played on March 18th, 2015. The watching took place in a controlled environment, so as to avoid situations where users would lose parts of the match because being distracted by other events around them. The audio used was in Spanish language. During the match, we asked the participants to annotate the main events and the time of occurrence. We did not limit the number of possible events. Also, participants were free to define what was an important event based on their own expectations. This helps capturing unusual events besides the common ones such as goals, shot on target/goal, faults, cards and substitutions. In total, the 7 participants created 77 different events for the watched match.

Separately, we created a summary visualization for the above match. In this visualization, we limited the number of video skims to 35, so they would properly fit the available screen size. From these 35 video skims found by our automated analysis, 29 correspond to events found by at least one participant when watching the raw video. This corresponds to 82.8% precision and 37.6% recall for our event detection. The low recall is mainly induced by two factors: (i) the subjectivity of the definition of important events – since participants can freely annotate the events, diversity is expected (in our study, 25 events were found by only one participant); and (ii) the limitation on the number of video skims imposed by the available display area – with more skims, better results could be attained, but this would increase the effort to interpret the visualization, which is not desirable. Regarding the events that all participants have considered as important, we had 62.5% precision – we show 10 of 16 events marked by all users. This indicates the good summarization abilities of our method with respect to the most important events.

After watching the match, the participants were instructed to use a prototype implementation of our interactive visualization. This phase started with a brief explanation of the visualization's goals and the available interaction mechanisms. After playing with the visualization tool, each participant answered a usability questionnaire. The statements included in the questionnaire (see Table 6.1) intend to measure the subjective level of engagement via a 7-point Likert scale, with 1 = strongly disagree and 7 = strongly agree (with the questionnaire statements). In total, the evaluation took around 130 minutes (100 for watching the video, 10 for learning to use the vizualization tool, 15 for using it, and 5 for filling in the questionnaire).

Q1	The visualization shows the most important match events.
Q2	The video skim sizes and colors faithfully reflect the real importance of an event.
Q3	The composition of video skims that represents a half time faithfully reflects the excitement/emotion of that time.
Q4	The overall visualization is a good match summarization.
Q5	The visual representation reflects the match quality.

Table 6.1: Evaluation: usability questionnaire.

Figure 6.8 depicts boxplots that summarize the participants' answers. Since we did not have enough participants to be able to compute statistical significance, we report only descriptive statistics. According to the answers for Q1, participants mostly agree with the statement that the most important events are captured by the visualization (6.43 on average). We already expect this outcome since, as described earlier, our technique showed a very good precision/recall on capturing the main events for this match. For Q2 (6.0 on average), the participants mostly agree that the size and color saturation of a video skim faithfully reflect the real importance of an event, confirming that the merge of audio+metadata highlights is a good approach to summarizing a match. Considering Q3 (5.71 on average) the participants do not fully agree with the size of the composition of video skims as a good way to reflect the excitement/emotion of a half-time. The visualization for this match presents compositions with similar sizes for both half-times. However, according to the participants, the second-half was more exciting. This is probably due to the fact that the goal occurs on the first half and the corresponding video skims are the larger ones on the visualization, thus approaching the sizes of the compositions. This can be overcome reducing the importance of a goal, or increasing the number of events to detect – we have tested increasing the number of events on the final visualization, and all new events are detected on the second half. Participant feedback for Q4 and Q5 is similar. They tend to agree that our visualization is a good summarization and that the visual representation reflects the quality of a match (6.71 and 6.43 on average, respectively), indicating that although a simple representation, our visualization method and tool can be a interesting alternative for match or tournament browsing.

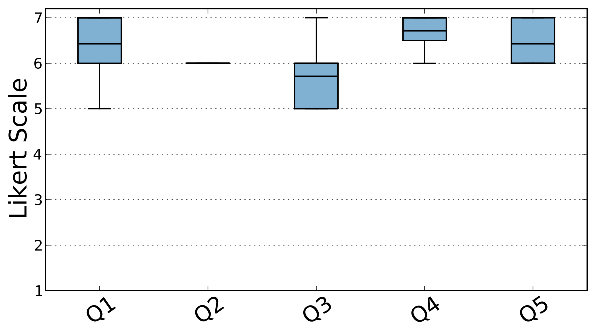


Figure 6.8: Statistics of users' feedback gathered via usability questionnaires.

6.5 data-centric approach: sports video exploration with multidimensional projections

As mentioned in Sections 6.1 and 6.2, a different way to look at information extracted from sports multimedia data is in terms of multidimensional datasets. As we have seen in the earlier chapters of this thesis, multidimensional projections are efficient and effective tools to visually explore such datasets. As such, several question arise naturally: Can we visually explore our sports multimedia data by using multidimensional projections? And, if so, which are the advantages and disadvantages of this type of exploration as opposed to the videoplayer-style exploration presented in Sections 6.3 and 6.4?

To answer the above two questions, we developed a second approach to explore our sports videos, by using a projection-based visualization. The dataset used for our visual exploration of sports videos using multidimensional projections is the same collection of soccer videos mentioned and analyzed so far, containing 8 matches (videos) from the 2014 FIFA Word Cup championship. This includes all the matches that the finalists played in the knockout stage plus the third-place match. The dataset contains games involving eight different teams, with the respective scores being the following: Germany 2 *vs* Algeria 1; Argentina 1 *vs* Switzerland 0; France 0 *vs* Germany 1; Argentina 1 *vs* Belgium 0; Brazil 1 *vs* Germany 7; Netherlands 0 *vs* Argentina 0; Germany 1 *vs* Argentina 0; Brazil 0 *vs* Netherlands 3. All the videos have a total of 55252 seconds (more than 15 hours of gameplay). From this dataset collection, we extract the same audio-and-metadata information as used by our first videoplayer-style exploration approach (Section 6.3). Based on this information, we construct a multidimensional dataset (Section 6.5.1).

As stated before, the central aim of our projection-based exploration is to see what types of insights can this approach deliver, and contrast these with the insights delivered by the first videoplayer-style exploration. When comparing these two approaches, usability issues of the two systems are obviously important to consider. Specifically, our videoplayer-based approach is arguably very simple to learn and use. As such, the projection-based exploration should also be as easy to learn and use as possible. This way, differences of the two exploration metaphors that are caused by usability issues will be diminished, so it is easier to compare these two metaphors. To achieve this, we aimed to design a projection-based exploration system which is as easy to operate by end users as possible, *i.e.*, contains a comprehensive graphics user interface with suitable menus and short-cuts, and various direct interaction and customization mechanisms. For this, we based our visual tool design and implementation on the VisPipeline framework, which is a comprehensive system for computing and exploring multidimensional projections [159]. VisPipeline allows an expert user to build a pipeline encompassing all the different aspects of visual exploration with multidimensionality projection techniques, including data clustering, a wide range of projection techniques, different visual encodings for the resulting 2D or 3D point cloud, and interactive selection and brushing mechanisms. Using this visual exploration tool, we developed and executed three exploration scenarios, each one having its own visualization design and types of considered data attributes (Section 6.5.2).

6.5.1 *Multidimensional Dataset Construction*

To reduce our video collection to a multidimensional dataset, we first need to define *observations* and their respective *attributes*. To define observations, we chose to encode in an observation a given time period of a given match, as described in more detail below. Given this model for an observation, its attributes are as follows (in line with the attributes employed in our first visualization design described in Section 6.3):

- *loudness/volume:* Describes the average audio volume for the timespan of an observation. Loudness is an ordinal-continuous attribute, computed with the RMS technique described in Section 6.3.1;

- *match_id:* Gives the ID of the match that an observation samples. This attribute is of nominal (categorical) type, and has eight distinct values, as our collection contains eight matches;

- *timestamp:* Gives the time (offset from the match start) at which a given observation was taken, measured in seconds. This attribute is of ordinal-discrete type;

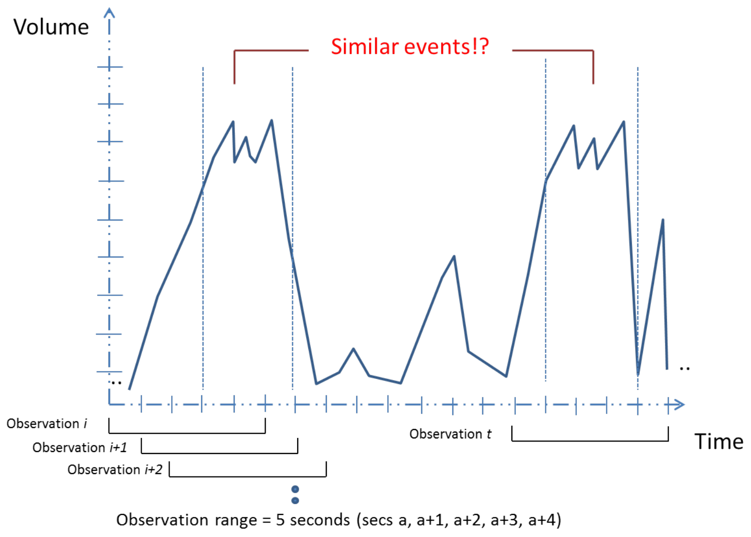
- *event_type:* Gives the type of event that the associated metadata indicates for a given observation. This attribute os of nominal (categorical) type, and has five values: 0=no event; 1=substitution; 2=yellow card; 3=red card; 4=goal. Note that, in the videos present in our collection, no red cards were given. However, we include the attribute here for completeness, as other soccer matches do, obviously, include red cards.

The construction of observations, or samples, from the raw video data follows a sliding-window approach, described next (see also Figure 6.9). Consider a time window of length Δt seconds. We slide this window over the entire duration of a video, with increments of $\delta t < \Delta t$ seconds. This delivers $T/\delta t$ positions of the window. For each such position, we compute an observation based on the information contained in the video material encompassed by the respective window. For instance, given such a time window, we compute the corresponding observation *loudness* value by using the RMS technique outlined in Section 6.3.1; the $timestamp$ represents the window's center; and the $event\_type$ describes the event taking place within the time window, if any. The sliding-window approach for transforming the video data into multidimensional data is justified by several points. First and foremost, our goal is to use next multidimensional projections to find similar and/or highly different fragments of the same and/or different match videos. As projections place similar observations close to each other, this gives a straightforward way to detect similar video fragments (by finding closely projected points) and also outlier video fragments (by finding projected points which are far away from most other points). Secondly, this approach allows us to control the scale, or level-of-detail, on which we analyze our multimedia data, by simply changing the Δt and δt parameters. In detail:

- Δt controls the *time-scale* at which we consider a single event to exist. Increasing Δt creates less observations, thus a more compact, easy to follow, and scalable visualization, but also filters out rapid events, which may be important. Also, too large Δt values may create time-ranges which contain more than one events as described by the metadata. Since $event\_type$ is a scalar attribute, taking a single value per time window, this will discard events. Decreasing Δt provides more fine-grained dynamic detail, but also creates more observations, which make the subsequent visualization slower, and increase the chances for observation clutter. A good setting for Δt is around the time-extent of the smallest *relevant* event in a soccer match, *e.g.* goal, fault, pass or corner, which is about 4 to 7 seconds.

- δt controls the *continuity* of the resulting projection-based visualization. If δt is small, then consecutive time windows will overlap significantly, thus their corresponding events will be very similar. Also, a larger number of events is created for a given video. This will make the resulting projection denser, since it has more observations *and* observations for consecutive time windows are more similar. Visually speaking, thus, such a projection looks more continuous, since there are more samples per unit area of the projection space, and these are also more similar to each other – we use in this reasoning the fact that a good projection, such as LAMP, preserves well similarities or distances between the high-dimensional and projection spaces, see Chapter 3. However, clutter is also increased by the larger number of observations. Conversely, if δt is large, consecutive time windows will overlap less, thus their corresponding events will be less similar, and fewer in total count. This makes the resulting projection sparser, and also more discontinuous, thus potentially harder to follow. If $\delta t > \Delta t$, then consecutive time windows do not overlap at all. The probability that they will have similar attributes consequently decreases, leading to observations which are less similar, thus more distant, in the resulting projections. Ultimately, this makes spotting trends in the resulting projections harder. Increasing δt too much, in cases where the window-size Δt is relatively small (a few seconds) has the additional adverse effect of reducing the probability that an audio event, seen as a time-range where the audio signal exhibits a high and sharp peak, will get 'cut' into two consecutive windows, thereby yielding too low RMS values to be detected. In practice, we observed that setting δt to about 20% of Δt gives a good balance between a reduced observation count and a good detection of rapid events in soccer videos.

Figure 6.9 shows the illustration of the proposed sliding window approach. In this case, we set $\Delta t = 5$ seconds and $\delta t = 1$ second, meaning that two consecutive time windows overlap for four seconds. To get a better understanding of the effect of these settings, we visualize the loudness signal (top graph in Fig. 6.9). We see here two large and quite similar loudness peaks, of roughly 3 seconds duration each, located at different time moments. These could indicate similar and/or important events.



Dataset

id	timestamp1	timestamp2	timestamp3	timestamp4	timestamp5	Abs_time	Event_type	Id_match
obs i	a	a+1	a+2	a+3	a+4	i/max_time	4 (goal)	7(final)
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.

Figure 6.9: Construction of a multidimensional dataset from a video collection.

A separate concern for the *loudness* attribute is normalization. This is needed since (a) average loudness may vary across different match videos, depending on the recording settings used by the respective content creators and/or the settings of the video codecs used to encode the final video data. To factor out such effects, we normalize the computed loudness values for each video to the same range. This design is based on the (reasonable) assumption that all matches contain equally-important lowest-loudness and highest-loudness values – or, in other words, that the loudness ratio between critical match events, such as goals, and irrelevant moments, such as low-action sequences, is roughly identical for all matches.

6.5.2 *Visual exploration of the multidimensional multimedia data*

After constructing our multidimensional dataset from the given video collection, we load the dataset into our VisPipeline-based exploration tool to generate and explore it via projections. As a projection technique, we use LAMP, which was extensively discussed in the earlier chapters. We consider both 2D and 3D projections, since both of these have their own advantages, as we have seen in

Chapters 4 and 5. As usual when exploring such projections, we use additional color mapping of a user-chosen attribute to the projected points to get more insight in the reason why close points are similar and faraway points are dissimilar. Additionally, we tune the Δt and δt parameters to achieve different insights in the underlying video data. The construction and interpretation of several such visualizations is described next by means of three exploration scenarios.

6.5.2.1 *Scenario 1: Using low-dimensional scatterplots*

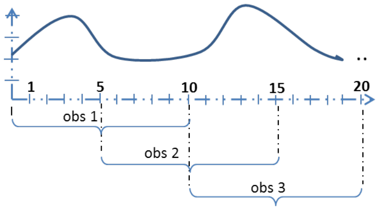
In this scenario, we essentially aim to determine the amount of insight in the match dynamics which we can find using a *low-dimensional* data representation. That is, we convert the multimedia data to a multidimensional dataset having a small number of dimensions. Next, we use LAMP to create scatterplots of this dataset, and explore them next.

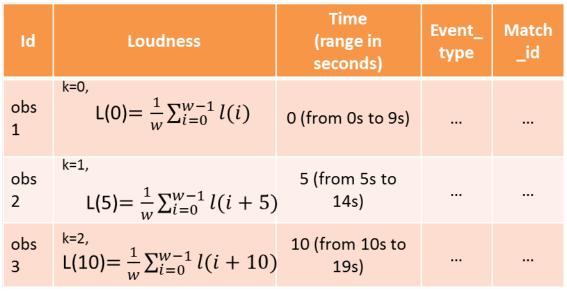
For this scenario, the multidimensional dataset obtained from the video data is composed by only four dimensions: *loudness*, *timestamp*, *event_type*, and *match_id*. We use here a setting of $\Delta t = 10$ seconds and $\delta t = 5$ seconds. This creates one observation for each 5 seconds of video data, yielding a total of 11043 observations.

The design of this multidimensional dataset is illustrated in Fig. 6.10. Here, the loudness $L(k)$ of the k^{th} 5-seconds interval in the video is the mean value of the next $w = \Delta t$ seconds

$$L(k) = \frac{1}{w} \sum_{i=0}^{w-1} l(i+k), \tag{6.4}$$

where $l(i)$ is the RMS loudness value for second i and $w = 10$ is the size of the time window (in seconds). As visible in the loudness time-graph (Fig. 6.10 left), the large time-window creates a smoothly-varying loudness signal in time.





Id	Loudness	Time (range in seconds)	Event_ type	Match _id
obs 1	k=0, $L(0)=\frac{1}{w}\sum_{i=0}^{w-1} l(i)$	0 (from 0s to 9s)	...	...
obs 2	k=1, $L(5)=\frac{1}{w}\sum_{i=0}^{w-1} l(i+5)$	5 (from 5s to 14s)	...	...
obs 3	k=2, $L(10)=\frac{1}{w}\sum_{i=0}^{w-1} l(i+10)$	10 (from 10s to 19s)	...	...

Figure 6.10: Dataset design for scenario 1.

Figure 6.11a shows the LAMP 2D projection of the multidimensional dataset constructed as described above, and rendered with fully opaque points. As visible, the result is quite cluttered, and no clear patterns are discernible. To alleviate this, we next render all points with a transparency of 40%. The result shows a more differentiated pattern (Fig. 6.11b). Here, we see that the central zone of the projection has a relatively higher observation density as compared to the peripheral regions, following a northwest-to-southeast gradient.

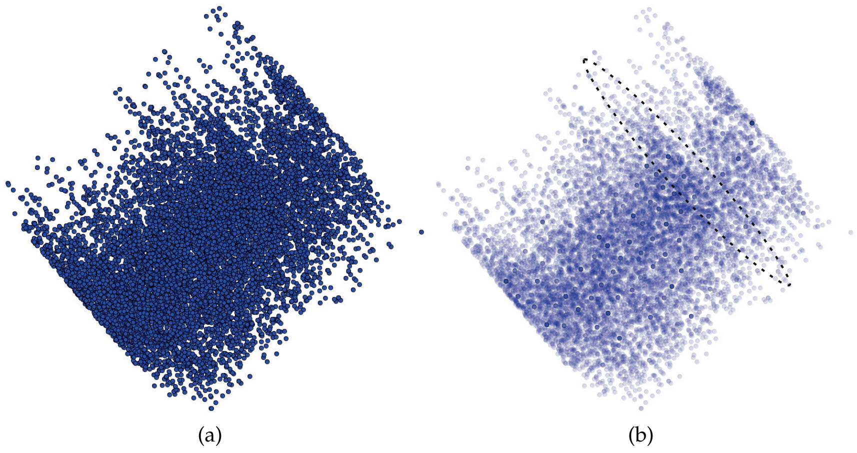
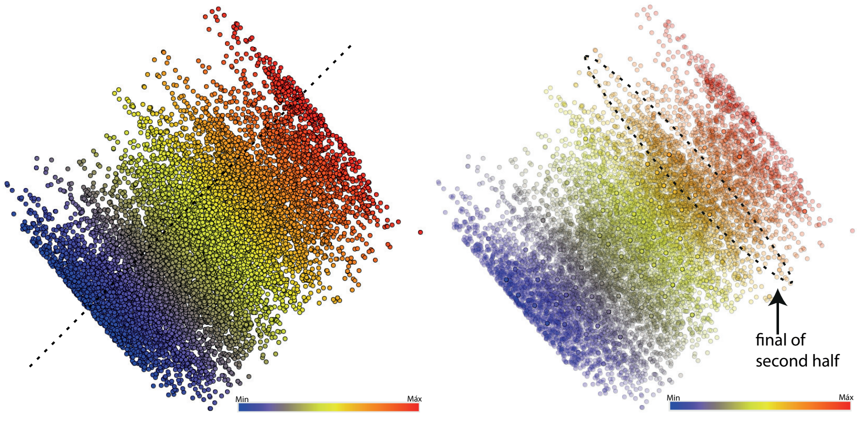


Figure 6.11: Scenario 1. (a) Point cloud. (b) Point cloud with 40% transparency.

Obviously, while the above-mentioned patterns may be interesting, they are (yet) hard to interpret. To improve this, we next color-code the projected points based on the value of a user-selected attribute, via a blue-yellow-green divergent colormap. Figure 6.12 shows the result when *timestamp* is the selected attribute. We see here an almost linear southwest-northeast color gradient, indicated by the dashed line in the figure. Figure 6.12b shows the same image, using a reduced 40% transparency, as in the previous figure. The interpretation of the projection is now simple: By exploring the color-coded plot with interactive brushing, we understand that the spike marked by the dotted ellipse in Fig. 6.11b indicates the time transition between the end of the second half of the visualized match (all points located southwest of the dotted ellipse) and the beginning of extra time (all points located northeast of the dotted ellipse).

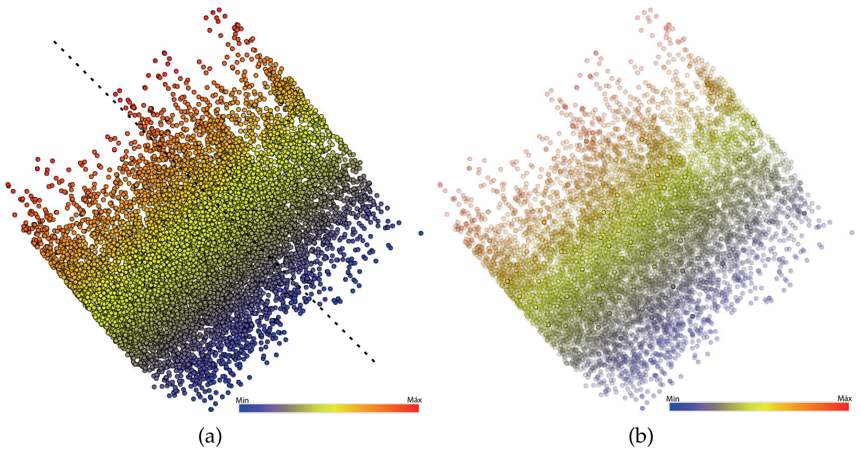
Figure 6.13a color codes the points by *loudness*. In contrast to the earlier images, we see now a color gradient going southeast to northwest, as indicated by the black dashed line. This gradient is almost perfectly perpendicular to the one observed when coloring points by *timestamp*. The two gradients basically explain the projection structure: Its diamond shape mainly follows two variables – *timestamp* (southwest-northeast) and *loudness* (southeast-northwest), see Fig. 6.14a further. Using a reduced transparency, we notice that most points are in the yellow area, meaning medium volume (Figure 6.13b). The red points in



(a) (b)

Figure 6.12: Scenario 1. (a) points color coded by timestamp. (b) points color coded by timestamp with 40% transparency.

this visualization represent the loudest audio samples in the video collection, possibly indicating interesting events.



(a) (b)

Figure 6.13: Scenario 1. (a) points color coded by loudness. (b) color coded by loudness with 40% transparency.

The projection visualizations discussed so far highlight two attributes only – *loudness* and *timestamp*. We next examine the *event_type* attribute. Since this is a categorical attribute, as described earlier, we cannot readily mix it with the quantitative *loudness* and *timestamp* attributes to create a projection. As such, we show next *event_type* using color coding (Fig. 6.14). Figure 6.14b shows the goal events as red points; the yellow cards as yellow points; and substitutions by gray points respectively. To further interpret the projection, we use the already understood meaning of the *loudness* and *timestamp* axes, shown in the projection as dotted lines. By following the correlation of the loudness axis with the color-mapped event types, we see that all goals are louder events (which is not surprising). Following the correlation of the time axis with event types, we notice a group of goals that happened in the beginning of the first half, and another group taking place in the extra time. On the northeast side of the visualization, we find a group of five goals in three different matches that occurred in extra time. Upon closer inspection, done by brushing, we find that these are the goal of the final match of Argentina *vs* Germany; the only goal of Argentina *vs* Switzerland; and all three goals of Germany *vs* Algeria. On the left side of the visualization, where goals occurred in the beginning of the first half, we see four more goals occurring in two matches, Brazil *vs* Germany and Argentina *vs* Belgium. As expected, in all these five matches, the goal points are surrounded by other points representing close observations (time-wise). This means that more than one each goal event is covered by more than one observation, as an effect of the sliding window approach.

Further interpretation of the correlation of the *loudness* axis with the color-coded *event_type* attribute shows us that most of the yellow cards are above the medium loudness, and most substitutions have medium volume, respectively. Separately, interpreting the correlation of the *timestamp* and *event_type* we discover that no substitutions happened in the beginning phase of the matches.

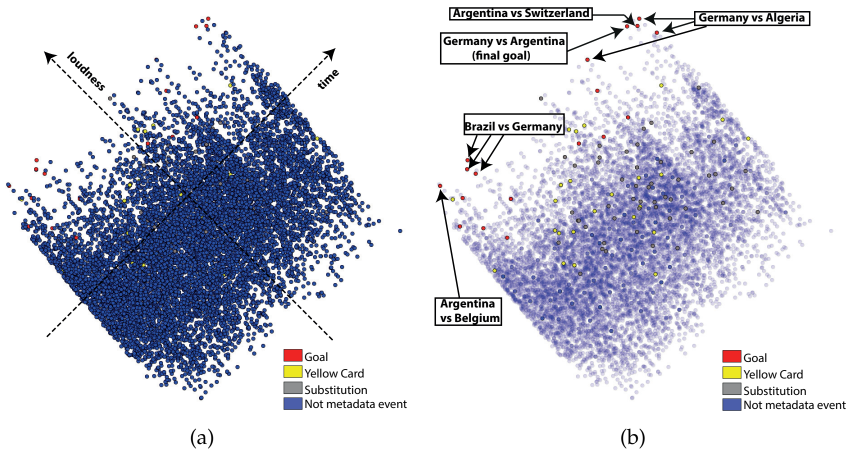


Figure 6.14: Scenario 1. (a) points color coded by event types, with *time* and *loudness* axes outlined. (b) points color coded by events with 40% transparency.

Figure 6.15a shows the same projection as in Fig. 6.14, this time color coded by *match_id*. To reduce occlusion, we use again transparency (Fig. 6.15b). This helps us discover an area in the projection which is less densely populated by observations – see dashed-line rectangle marker in Fig. 6.15b. This indicates that only a subset of the studied matches have observations related to this area. As already found in the time analysis (Fig. 6.14a), this zone corresponds to the end time-range of matches, and more specifically, to extra time. The fact that there are fewer samples here is in line with the fact that only four matches in our dataset have extra time. We recognize these also by the colors of the observations in the dashed area: Germany *vs* Algeria (blue points), Argentina *vs* Switzerland (green points), Netherlands *vs* Argentina (red points) and Germany *vs* Argentina (orange points). Note, also, that the assessment of the loudness distribution is harder to make in Fig. 6.15b as compared to Fig. 6.14b. This is due to the fact that the *match_id* attribute has a much flatter distribution over the sample set than the *event_type* attribute – most points have *event_type=none*. In turn, this creates a more complex color pattern in Fig. 6.15b, which adversely interacts with blending, as well-known in data visualization [217].

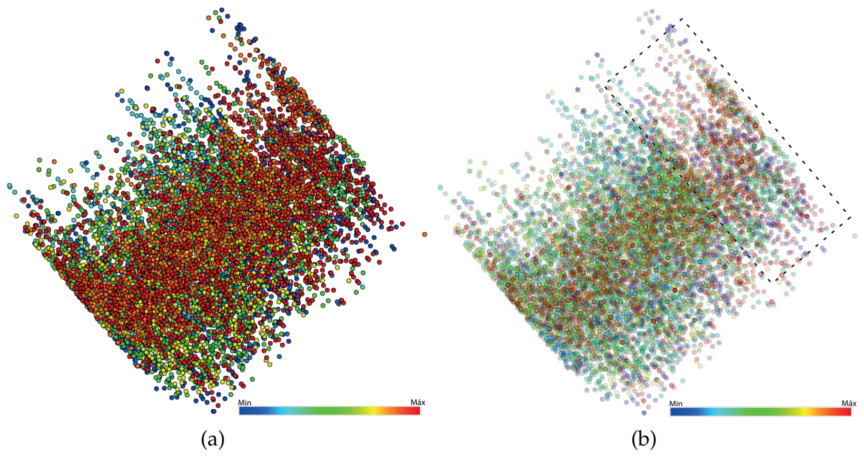


Figure 6.15: Scenario 1. (a) points color coded by *match_id*. (b) points color coded by *match_id* with 40% transparency.

6.5.2.2 *Scenario 2: Increasing the number of dimensions*

As described in Section 6.5.2, an interesting use-case is finding similar events based on the loudness of the audio signal in a given time window. In the previous scenario, we explored the data by computing a single loudness value L for each such time window (Eqn. 6.4). While this reduces the number of dimensions needed to represent the data, it also considerably smooths out the audio information, as discussed in the beginning of Sec. 6.5.1. Also, using a single audio dimension per window makes observations too unspecific, while our final intention is to be able to compare multimedia *patterns*. What we need, is more specific, thus higher-dimensional, observations.

To achieve this, we next encode the audio information of a time window (observation) using several dimensions. For this, we use a sliding time window of $\Delta t = 5$ seconds, slided by $\delta t = 1$ second. This offers a good time resolution, as discussed earlier. Next, we create five dimensions $l_i(k)$, $1 \leqslant i \leqslant \Delta t$ from the audio data of every window k, each corresponding to the RMS loudness of the i^{th} second in the window. Hence, an observation $\mathbf{x}_k$ will have eight dimensions

$$\mathbf{x}_k = (l_0(k), l_1(k), l_2(k), l_3(k), l_4(k), time(k), event\_type(k), match\_id(k)). \tag{6.5}$$

As such, the complete multidimensional dataset for scenario 2 has 55252 eight-dimensional observations. We next explore this dataset by projecting it to 2D using LAMP and encoding different attributes to color, as in the first scenario described in Sec. 6.5.2.1.

Figure 6.16 shows the dataset projected by considering the loudness dimensions $l_i(k)$, *timestamp*, and *match_id*. Points are colored by *event_type* and rendered half-transparent. The resulting pattern is quite similar to the one shown for scenario 1 (Sec. 6.5.2.1), apart from being rotated. As such, it appears that increasing the resolution used to capture loudness from one dimension to five dimensions does not reveal additional patterns. However, this impression may be due to the fact that the *event_type* and *match_id* variations dominate the loudness ones, and as such, are the main drivers that influence the projection's shape.

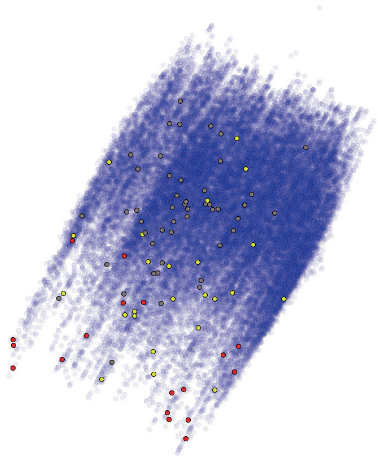


Figure 6.16: Scenario 2 with 7 attributes per observation (five loudness, *event_type*, and *match_id*)

.

We believe that the loudness dynamics captures more information than the previous visualizations can show. One way to emphasize such dynamics is to project the data based solely on them. Figure 6.17a displays this, which shows an elliptical-like pattern surrounded by a few outliers. Again, to reduce occlusion, we use transparency (Fig. 6.17b). It shows better how is the distribution of loudness over time, having most of the points placed on the right side of the flat circle and a kind of a spike on the left side. While this basic visualization does not (yet) tell us more information than the ones presented in Sec. 6.5.2.1, its significantly different pattern tells us that there is a good chance that this projection can reveal different facts.

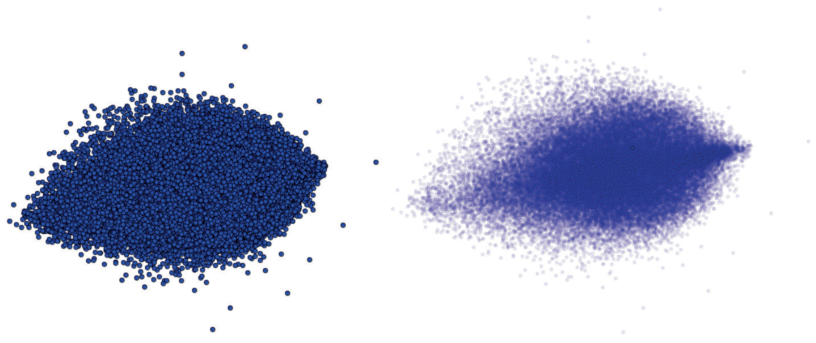


Figure 6.17: Scenario 2. (a) point cloud, projected by 5 audio dimensions. (b) point cloud with 40% transparency.

To explain why the shape of the projection in Fig. 6.17 supports our claim that loudness dynamics contains interesting patterns, consider the following analysis. Suppose that all loudness patterns for all observations would be spread randomly, and uniformly, over their five-dimensional space given by attributes $l_0, \ldots, l_4$. This would in turn imply that the LAMP projection of this dataset would be roughly a 2D uniformly-sampled disk [31]. Clearly, our projection's shape deviates from a disk, and also does not exhibit an uniform density. Hence, the loudness patterns are distributed non-uniformly in high-dimensional space, so they potentially contain interesting information.

To make this projection useful, we next color map the dimensions not used by the projection. For this, we first use *event_type*. Additionally, we make observations having no relevant event information (*event_type=none*) half-transparent, to emphasize the eventful obervations. We now notice that most events are located in the left side of the projection (Fig. 6.18a). A zoom-in of this image (Fig. 6.18b) shows there are more red points (goals) left of the zoomed-in area; yellow points (yellow cards) are spread rather uniformly over the image; and gray points (substitutions) more concentrated on the right side. Since the projection encodes audio *patterns*, using five dimensions per observation, this means that the audio patterns related to goals are most different from the ones related to substitutions, while the audio patterns for yellow cards are broady similar to both goals and substitutions. This observation confirms standard knowledge about soccer games: Goals typically create salient audio patterns, as they are the most important match events; substitutions are typically quied match moments, very different from goals; while (yellow) cards cover the entire spectrum of calm to high action, depending on the reason for substitution and the preference of the commentator and/or predominant stadium spectators for the player involved in the card. Additionally, the fact that most events, and in particular the critical ones (goals) are located left in the projection, tells us that the horizontal projection axis can be interpreted as overall loudness (left=loud, right=silent), a finding which can be easily checked by *e.g.* brushing the observations.

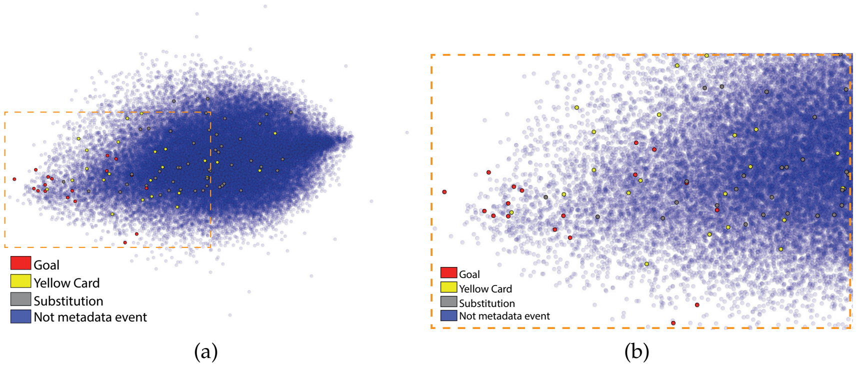


Figure 6.18: Scenario 2. (a) point cloud, projected by 5 audio dimensions. (b) zoom-in detail with points colored by *event_type*.

Another question is whether events, loudness patterns, or the dynamics of different matches correlate in some significant way with time. To explore this, we color the above projection by *timestamp*. The result is shown in Fig. 6.19a, using a divergent three-color colormap (blue=match start; yellow=half time; red=match end). The projection's middle area appears predominantly red. However, using transparency shows that this is just an effect of occlusion (Fig. 6.19b). In other words, five-second audio-dynamics patterns are not significantly correlated with the moment in time they appear during a match. However, the opaque projection rendering shows us several salient yellow outliers, *i.e.*, match moments whose audio patterns significantly differ from the rest, and are mostly located around half time (Fig. 6.19a).

To further make sense of the outliers, we colored them by the *match_id* dimension, as this dimension was not yet studied in the current analysis (Fig. 6.20a). We now see that most orange outliers located to the right side of the projection, indicated by black arrows in Fig. 6.20a, belong to the same match. Orange corresponds to the match Brazil *vs* Netherlands, playing for the third place. Brushing

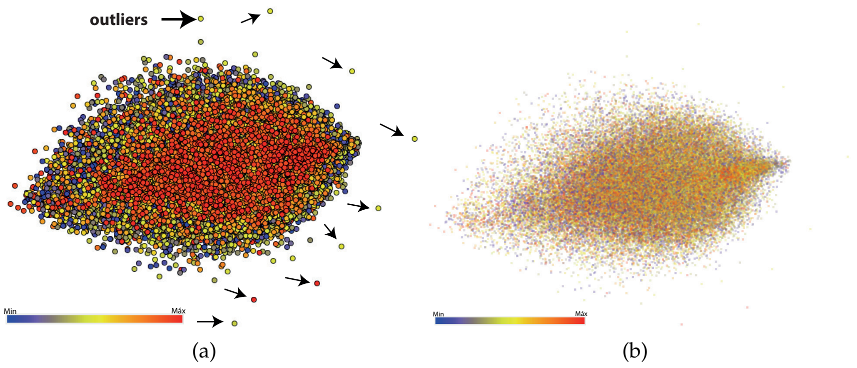


Figure 6.19: Scenario 2. (a) points color coded by *timestamp*. (b) points color coded by *timestamp* with 40% transparency.

the points we discover they all correspond to neighbor time moments in that match. Watching the actual match video, we discover the reason for these outliers: They all correspond to a silence event occurring during that match, related to a transmission problem of the broadcaster. Similarly, the two blue points, indicated by red arrows in Fig. 6.20a, are easy to trace to the match Germany *vs* Algeria. Watching the video shows that they also correspond to the same type of silent events. As such, we have explained projection outliers in terms of data recording problems. Knowing this, one can eliminate these outliers to proceed with an examination of the relevant, reliable, data.

Alternatively, we may be actually interested in the outliers, and wish to understand them better. To do so, we further explore the 2D projection shown in Fig. 6.20a. We next zoom on the left tail of the projection pattern (black rectangle) and obtain Fig. 6.20b. We can now see easily that the loudest points, indicated by arrows in the figure, are red. This corresponds to match 6 between Germany and Argentina. Brushing these points to find their exact timestamps, and examining the videos at those timestamps, shows that these observations correspond to Germany's goal in that match, which is the loudest event in our entire dataset.

A further possibility to see more structure in the projected data is to use a higher-dimensional projection than two dimensions. Figure 6.21 shows this approach, where we used a 3D LAMP projection, which is further explored using the interactive tools presented in Chapter 5. In the projection, we see five biplot axes, corresponding to the five loudness attributes $l_i(k)$ in Eqn. 6.5. The five green icons under the x axis legend (arrow in Fig. 6.21) show that the loudness increase is from left to right.

By looking at the green boxes of the x axis legend (pointed by a black arrow), we can infer that the positive loudness direction is from left to right. As such, we can say that the points on the far right side of this figure represent the loudest volume and the points on the far left side represent the lowest volume. The silent

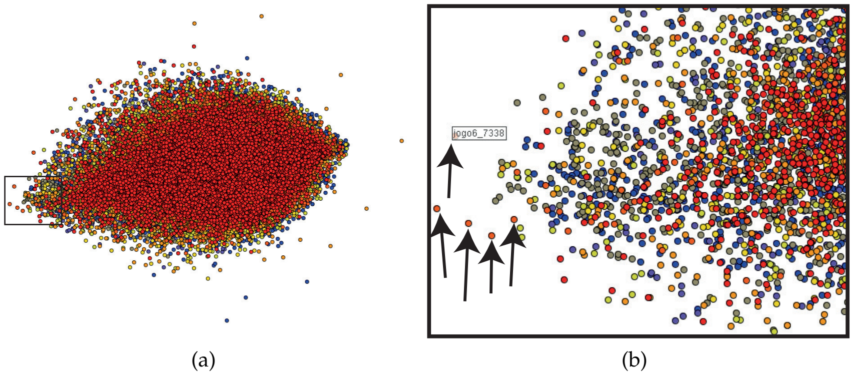


Figure 6.20: Scenario 2. (a) Points colored by *match_id*. Orange and blue points represent silent events. (b) Zoom-in on the high-loudness outliers.

events located on the outside border of the point cloud can be also explained by the biplot axes in the sense they are on the left side of the figure. As such, the right 'tail' of the produced 3D point cloud is describing the loudest events.

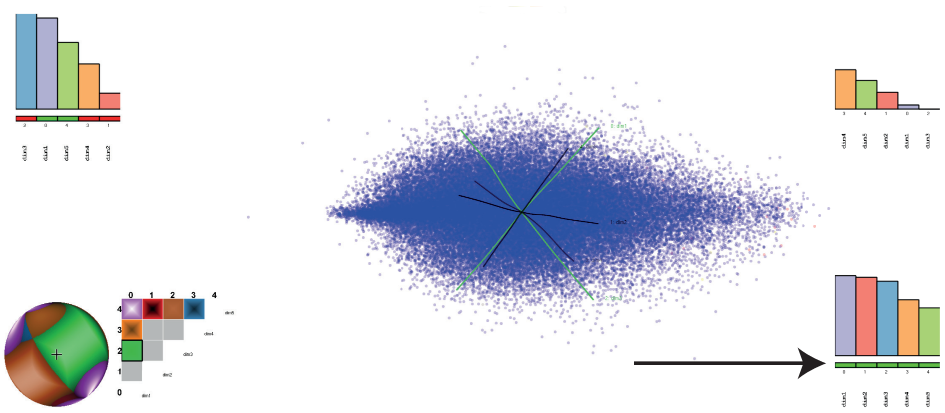


Figure 6.21: Scenario 2. Data projected in 3D using five *loudness* dimensions.

We can also use 3D projections, including biplot axes, to understand how time is spread over the observations. Figure 6.22) shows a 3D LAMP projection of the six-dimensional dataset including the already mentioned five loudness dimensions and the timestamp dimension. In this figure, we also use the axis legends to align the *timestamp* axis with the x screen axis, by the mechanisms described in Sec. 5.3.2, and also color points by *timestamp*, for a clearer view. By slightly rotating the 3D projection, we realize that virtually all five loudness axes are strongly correlated (have small angles), so we next align them with the y axis. We can now easily see that the globally loudest observations are red, thus occur in the final match phases (see top-right marker in Fig. 6.22). Similarly, we see that the most silent observations occur roughly during half time (bottom marker in Fig. 6.22).

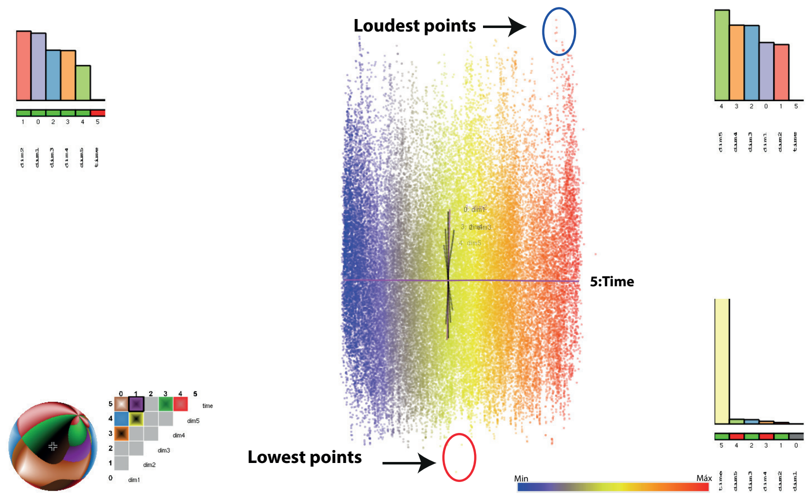


Figure 6.22: Scenario 2. Data projected in 3D using five *loudness* dimensions and *timestamp*. Color codes *timestamp*.

6.5.2.3 *Scenario 3: Further increasing dimensionality*

As scenario 2 showed, increasing the number of dimensions used to represent our multimedia data (in particular, the audio information) can be useful to detect more subtle patterns and outliers as when using a small number of dimensions (scenario 1). We now refine this finding by increasing the number of dimensions even further.

To do this, we use the same procedure as described in Sec. 6.5.2.2. We use a time window of $\Delta t = 10$ seconds slided by $\delta t = 5$ seconds, similarly to scenario 1. We next compute the RMS loudness $l_i(k)$, $1 \leqslant i \leqslant 10$, of each of the ten seconds in this window, similarly to scenario 2. Adding the *timestamp*, *event_type*, and *match_id* attributes, this yields 13 dimensions per observation, for a total of 11045 observations.

Similarly to Fig. 6.16, Figure 6.23 shows the dataset of scenario 3 using ten loudness dimensions and the timestamp dimension as input for a 2D LAMP projection. Points are colored by *event_type*, and rendered with 40% transparency. As visible, this projection has the same overall shape as the ones obtained using one loudness dimension plus *timestamp* (Fig. 6.14) or five loudness dimensions plus *event_type* and *match_id* (Fig. 6.16). This strenghtens our earlier-outlined observations that mixing loudness, timestamp, and match ID attributes in the same projection is not a good idea, given their very different natures. For completeness, we note that the LAMP implementation we use does normalize the variables, so that they contribute equally to the inter-observation distances.

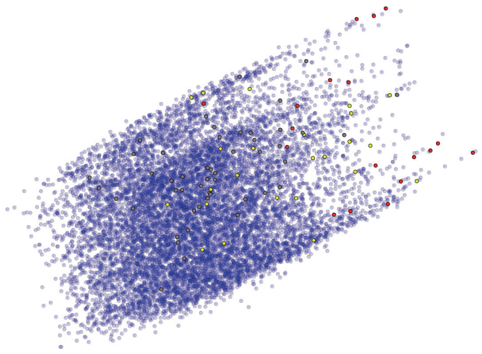


Figure 6.23: Scenario 3 with 10 loudness and one *timestamp* dimensions per observation, colored by *event_type*.

To be able to explore all above-mentioned variables, we use the same procedure described in Section 6.5.2.2. In detail, we project the data to 2D using only the ten loudness dimensions, and next explore their correlation with the remaining dimensions using color coding. The 2D projection of the data using the ten loudness dimensions is shown in Fig. 6.24a. The overall shape of this projection, and also the presence of a few outlier observations close to its border, is very similar to Fig 6.17a, which only used five audio dimensions. The same is seen when comparing the 2D projection of our ten-dimensional dataset (Fig. 6.24b) with the similarly-obtained 2D projection of our five-dimensional dataset (Fig. 6.17b). Additionally, color coding in Fig. 6.17b by *event_type*, shows us that most goals are located in the bottom part of the projection. If we use the already established correlation of goal events with high loudness, we can thus infer that the bottom part of the projection corresponds, roughly, to high loudness observations. By using the biplots again, Figure 6.25 confirms that the loudest samples are indeed in the above-mentioned area – indeed, all green boxes in the y axis biplot indicate that attribute values increase from bottom to top in the projection. Since boxes for *all* biplot axes are green in the y legend, this means that all our ten loudness values increase from top-to-bottom; in other words, samples at the top of the projection are loudest over the *entire* extent of their time windows. However, we see that the biplot axes of these dimensions, shown in black at the center of the projections, are *not* parallel. This means that our 10 loudness dimensions are not strongly correlated or, in other words, that the considered loudness patterns are different.

The similarity of Figs. 6.24 and 6.17 indicate that the captured loudness patterns are similar on both time-scales used to analyze the datasets (sliding window size of 5, respectively 10 seconds). This is desired, since it tells us that our findings obtained by such projections are not significantly influenced by the choice of the time-window size. In other words, such patterns are present at several *scales*.

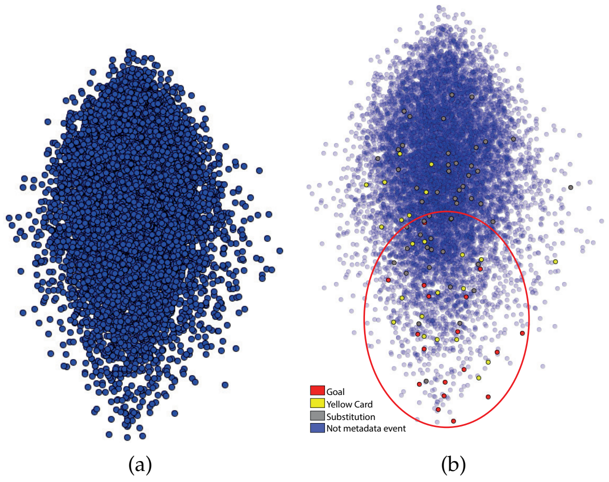


Figure 6.24: Scenario 3. (a) points projected using 10 loudness dimensions. (b) same plot, color coded by *event_type*, and rendered with 40% transparency.

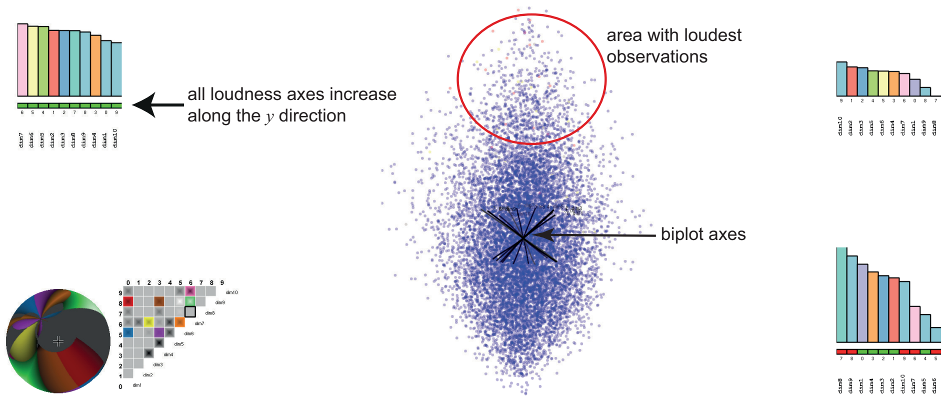


Figure 6.25: Biplots of Scenario 3 showing correlations of goals with loudness and variability of the loudness patterns.

Similarly to Scenario 2, we next use the biplots to understand how time is spread over the observations, and how it correlates with the 10 loudness dimensions. For this, we project the 11-dimensional dataset formed by the loudness dimensions and time. Next, we align the time axis with the x screen axis, by clicking in its bar in the x axis legend. For extra emphasis, we color points on their time value. Figure 6.26 shows the result. Several observations can be made here, as follows. First, we see how the biplot axes for all loudness dimensions are almost orthogonal to the time axis. This tells that time and loudness are not strongly correlated, *i.e.*, we have roughly similar loudness ranges for all match phases (time ranges). More interestingly, this view strongly resembles Figures 6.16 and 6.23, which were 2D projections using loudness and time. This is a good example of how specific viewpoints of 3D projections can yield insights obtained with 2D projections. The added-value of a 3D projection is, however, the possibility of generating a very large number of such 2D projections just by changing the viewpoint, thus with no need to recompute the projection (Chapter 5).

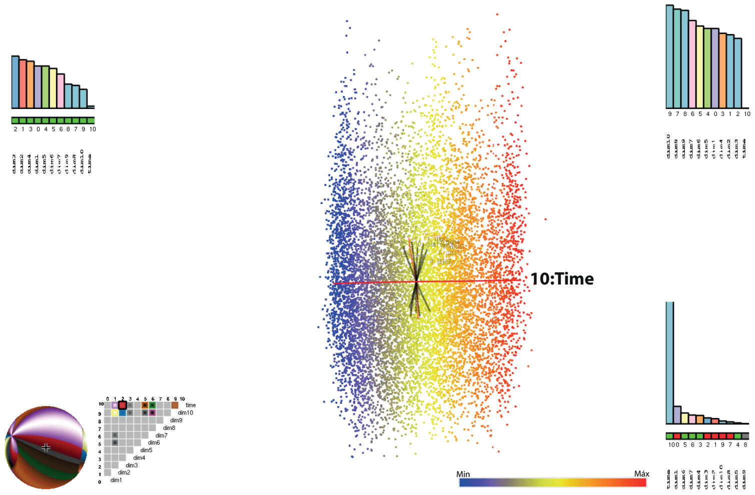


Figure 6.26: Scenario 3 showing loudness *vs* time correlation. Time is aligned along the x axis. Points are color coded by time.

6.6 DISCUSSION

Several points concerning our two approaches to visually exploring soccer multimedia data are relevant to discuss, as follows.

Data attributes: In our work, we extracted four 'raw' attributes from our soccer multimedia collection – loudness, match ID, timestamp, and types of events. These were further refined to produce a variable number of attributes used to construct the final visualizations – up to 13 for out multidimensional approach (Section 6.5.2.3). Obviously, many more techniques exist for extracting additional kinds of attributes from sports multimedia data (Sec. 6.2). Some of these, *e.g.* player motion extraction, are considerably more advanced than our relatively simple loudness-time-metadata analysis. However, open access to the implementation of such techniques and/or to the datasets they deliver is rare. Additionally, by using our simple attributes, which admit a simple interpretation and visualization, we can focus more easily on answering our research questions outlined in Sec. 6.1.

User-centered exploration: Our first visual design using the videoplayer-like metaphor shows that it is possible to construct an easy to learn and use tool for visual summarization, browsing, and comparison of real-world collections of soccer matches. The tool can effectively provide high-level insight into the salient phases and/or events of a soccer match; more interestingly, it allows an easy way to browse a large match collection, supporting tasks such as finding matches in which a certain pattern occurs, or comparing different matches. While our practical evaluation of this tool was limited to a small user group, the obtained feedback shows that this visual metaphor has a good potential. However, this visual metaphor requires a non-negligible amount of user interaction (browsing), and is arguably not suitable for fine-grained analyses.

Data-centered exploration: Our second visual design using multidimensional projections shows that such projections are useful in finding and interpreting correlations, patterns, and outliers in our sports multimedia data. Such datasets, which are inherently time-dependent, can be reduced in several ways to static visualizations, whose interactive exploration shows several interesting insights at different time-scales. As a side effect, we have seen that 3D projections, when augmented with our interactive exploration tools discussed in Chapter 5, bring added-value as opposed to static 2D projections. This added value will arguably be much larger for use-cases where many more dimensions are extracted from the multimedia data. However, data-centric exploration using projections is significantly more abstract than the videoplayer-based metaphors, and may be suitable only for specialized users. Separately, in all our three scenarios involving projections, we noticed the difficulty in *explaining* patterns present in projections, which stands in contrast to the easy interpretation of a time-range or event in the videoplayer-based approach. This is an important limitation of projections in general, which is an important topic for future research.

Limitations: Technically speaking, several points exist that can be further refined. For instance, depending on the narrator's excitement, loud time-ranges describing salient events such as goals may last more than one minute. This would add multiple consecutive videos skims related to the same event in both our visualizations – in other words, the event will be redundantly supersampled. Cross-match language-dependent normalization can help here in reducing such redundancy. Separately, if the match ends at the extra-time second-half in a draw, and penalty shoot-outs follow, our visualization can get biased by the excessive amount of metadata and/or audio intensity for that period. However, on the other hand, if a visualization shows a highly important set of end-match skims, this naturally implies that such events are (the most) important for the respective match, possibly dominating other events. Last but not least, arguably the most important limitation of our analysis is the limited evaluation of the proposed visual techniques. To gain more confidence regarding both the ease-of-use and acceptation, a large-scale study involving both casual and expert users is needed for both our visual designs. However, we note that such large-scale evaluations are very expensive to conduct, which is one of the causes why they are not present in the literature describing most video summarization applications, and are actually relatively sparse in visual analytics literature at large.

6.7 conclusions

In this chapter, we have studied the problem of constructing visual tools for the exploration of collections of soccer multimedia datasets. At a high level, the tasks that such tools aim to support are relatively easy to describe in terms of getting various insights into outliers, trends, correlations, and patterns involving the dynamics of the game. However, beyond this level, exploration strategies and supporting tools are widely different in terms of the type of data they consider, type of explorations they support, and type of users they are aimed at.

In our study, we considered two extremes of the above-mentioned tool-and-exploration spectrum. At one extreme, we consider casual users such as sports fans who are interested to quickly and easily browse a soccer match video, or a collection of such matches, to identify and get insights on high-level events of interest having taken place during the game, and also to compare the evolution of several such games, *e.g.* in a tournament. At the other extreme, we consider technical experts who are interested to study the patterns captured by the multidimensional attributes that can be extracted from a sports video collection. For both cases, we first analyze both audio and metadata streams associated with a match, and reduce these to sequences of important events, annotated by the event type, corresponding video fragments that illustrate the event, and other attributes of importance for understanding the event. Next, for casual users, we propose a videoplayer-like visual design that does not require learning complex visual or interaction metaphors. Hereby, casual users can quickly explore a summarization of one or several matches. We demonstrate the added-value and way of working of our proposal by analyzing several matches from the 2014 World Cup knock-out phase. Additionally, to assess the usability and perceived effectiveness of our proposal, we performed a user evaluation of the perceived quality and ease-of-use of our visualization tool. For expert users, we explore the extracted data using 2D and 3D multidimensional projections supported by interactive exploration tools. In contrast to the first approach, projections offer a finer-grained insight at observation level, scale better with respect to observation and dimension count, but are more abstract and harder to use.

Given the obtained insights, several directions for future work can be envisaged. On the technical side, our data extraction can be enhanced to handle broadcast transmissions systems via multimedia streaming. This would enable us to capture (and analyze) not only the audio of the real-time video broadcast, but also the dynamically generated metadata provided by sports websites. In turn, this would provide the type of high-level insight offered by our videoplayer-based metaphor in (near) real-time, allowing sports fans to *e.g.* compare matches and match events live, or analyze an ongoing match from the perspectives of multiple broadcasters. Separately, our work shows that projections are, technically, flexible and scalable enough to handle large volumes of multidimensional data extracted from multimedia collections. However, their use by casual end-users is still severely hampered by their abstract nature. This advocates for more research in *explaining* projections with *e.g.* automatic clustering and annotations. Finally, combining the complementary strengths of the two visual approaches described here (videoplayer-like and projection-based metaphors) could lead to novel, scalable, and easy-to-use solutions for the interactive exploration of large multimedia collections aimed at a wide spectrum of users, and with applications beyond sports videos.

7 DISCUSSION AND CONCLUSIONS

This thesis has presented visualization methods to interactively explore multidimensional datasets aimed from specialized to casual users, by making use of both static and dynamic representations created by multidimensional projections. In this final chapter, we revisit our initial research questions, stated in Section 1.5, discuss and compare the obtained results against these questions. We reflect on the completeness of our results, and further outline potential directions for future work.

First, let us revisit our research questions:

Question 1: *How can we design ways to interactively explore multidimensional projections that convey to users insights on the semantics of the patterns perceived in the projection space, in terms of aspects of the high-dimensional data?*

To answer this question, we first need a high-quality multidimensional projection technique, *i.e.* a technique which faithfully preserves distances, can generically handle any type of high-dimensional data, is scalable in both observation and dimension count, and is easy to use. The LAMP technique proposed in Chapter 3 largely meets these requirements, and is, as such, one of the main projection techniques used in the remainder of this thesis. The success of LAMP is also noticeable beyond the scope of this thesis, and is reflected in its being mentioned and compared against in several papers by various authors.

Once we have LAMP, or any other suitable projection technique, we need to assess its accuracy. This needs to be done before actually using the projection to create scatterplots which are next examined to reason about data patterns. Indeed, if the projection inaccurately projects (parts of) the observation set, then the corresponding projected points will create patterns which are wrong and can be misleading. Chapter 4 focuses on the exploration of projection errors. After studying the existing error metrics in the literature, we notice the need for more detailed metrics able to highlight specific types of projection errors on specific subsets of observations. To address this need, we introduce several types of such specific error metrics – aggregated, false neighbors, missing neighbors, false group members, and missing group neighbors. To explore these errors, we introduce several visual metaphors based on scalable space-filling techniques such as image synthesis, shaded cushions, and edge bundles. Using these techniques, we showed how different projection techniques can be compared against each other, and also characterized in terms of their behavior upon parameter change, and performed such analyses on several real-world datasets. This let us make several observations with respect to optimal settings for these techniques.

Such settings are next useful for constructing accurate projections for practical data exploration scenarios.

Having now projection techniques configured (and tested) to deliver high accuracy, we turn to the joint questions of explaining projections and deciding upon the usage of a 2D *vs* a 3D projection. On the one hand, 3D projections typically create lower projection errors, as compared to 2D projections (Chapter 5). On the other hand, 3D projections yield 3D point clouds, which are considerably harder to understand. We address the latter issue by proposing a number of interactive visual tools for explaining such projections. By using a set of interactive linked views and legends, we show how to explain the meaning of projected dimensions in terms of original variables; show projection nonlinearities and correlations (or lack thereof) for these variables; help finding good viewpoints from which given variable-pairs can be best explored; and quickly show which variable-pairs can be explored from any possible viewpoint. All of the previous insights significantly help the user to enhancing his/her semantic interpretation of a dataset. Separately, we use the error-exploration techniques introduced in Chapter 4 to compare 2D and 3D projections, and outline the advantages of the latter on a number of concrete questions related to both real-world and synthetic datasets.

Throughout the entire thesis, the proposed interactive exploratory tools for 2D and 3D projections are illustrated on numerous examples involving different types of multidimensional datasets emerging from simulation, physical measurements, software quality analysis, and surveys. The proposed tools serve a wide spectrum of tasks, ranging from technical ones, *e.g.* finding outliers, correlations of variables, and subsets of observations affected by specific projection errors, to more complex high-level data explaining tasks, such as identifying outliers and patterns formed by observation groups and explaining them, and identifying stable and unstable patterns upon projection-technique or technique-parameter changes.

Reflecting at a high level on the completeness of the answers provided to our first research question reveals a few important points.

First and foremost, projection errors (of various kinds and extents) are unavoidable by any projection technique, including the latest state-of-the-art ones, when treating high-dimensional real-world datasets. This state of affairs is very likely to exist in the future too, since certain high-dimensional configurations can hardly be mapped to low-dimensional spaces without having certain distance distortions. However, the effect of such errors upon actual user *tasks* is far less clear. In certain situations, such as localizing groups of strongly-similar observations or reasoning about the correlation of variables over specific subsets of points, having certain observations influenced by even very high errors will not influence the visual interpretation of the data. The crucial aspect here is the spread of errors over the observation set. Our proposed error metrics and related visualizations handle a number of basic problems of projections. However, for specific applications, where one has a clear understanding of particular types of problems emerging from errors, designing customized error metrics and visualization thereof should be studied. This is an open area for future research.

Secondly, our experience with 3D projections casts a new light on their practical usability and added-value. Contrary to several literature studies which argue that 3D projections are of little added value as compared to their 2D counterparts, we have found that such projections can serve many data-exploration tasks better than 2D projections, if augmented by suitable interactive explanation tools. Examples of such tasks are finding and reasoning about correlations of variables and finding and explaining local patterns created by observation subsets in terms of the original variables. Our explanatory tools essentially remove several of the disadvantages of 3D projections and thereby leverage their main advantage – lower projection errors. However, our current results are far from being able to remove all difficulties related to exploring 3D projections. Issues such as occlusion, and selecting point subsets for local exploration and explanation, are still open to examination.

Question 2: *How can we design ways to interactively explore multidimensional data extracted from multimedia datasets so as to support a wide range of tasks for different types of users ranging from professionals to casual users?*

We approach this question at two main points in the thesis. First, we propose in Chapter 3 a way to dynamically modify a multidimensional projection according to user knowledge. The underlying idea is that a projection will likely inaccurately reflect the *intended* similarity comparison of observations, due to several factors – the incomplete descriptive power of the original data dimensions (measurements do not precisely reflect what the user intends to compare in the data), imperfections in measuring these dimensions (measurements include estimation errors), and distance-preservation errors of the projection itself. As such, in cases where the user is able to specify the intended amount of similarity between specific observations, we propose to use this information to modulate the entire projection pipeline. In practice, this allows users to 'organize' a dataset by manipulating it in projection space to best reflect his/her pre-existing knowledge. Once this is done for a small number of observations, called control points, the entire projection adapts to arrange the remaining observations following these user-supplied constraints. While this technique is generic, its specific added value is demonstrated in the context of semantic exploration and organization of multimedia-related datasets, such as collections of images, video, and text. The tool resulting from this research offers a very intuitive and easy-to-use way to explore and organize such datasets by simply moving visual depictions of the observations in a 2D space, and can as such be used in a wide set of contexts involving both expert and casual users.

Secondly, we study ways to explore multidimensional multimedia datasets in the more specific context of video collections (Chapter 6). We follow here the approach taken in our first exploration of multimedia data of addressing the needs of both expert and casual users. To this end, soccer video collections offer a good test field: These are interesting both for experts, *e.g.* coaches and sports analysts, but also for casual users, such as soccer fans. To study solutions suitable for both these user groups, we take here a different approach than in Chapter 3, by designing two conceptually different visualizations. For casual users, we propose a videoplayer-like metaphor that allows one to quickly and easily browse a match video, or a collection of such matches (a tournament) to quickly detect important events and/or temporal patterns formed by such events, and next compare such patterns. We assessed the ease of use of the proposed visual metaphor by applying it to the matches from the final stage of the 2014 World Cup, and discovering by this usage interesting insights concerning different matches and different broadcast languages. To further evaluate our metaphor's ease of use and interest, we performed a user evaluation on the target group (soccer fans). The results of this evaluation confirm the ability of the tool to support finding and explaining important events in a match, and to support the speeding up of browsing video matches to get an overview of the match's high-level contents. For more experienced users, we studied the suitability of classical 2D and 3D multidimensional projections to support tasks such as identifying outliers, high-level correlations of the dimensions extracted from the raw video-and-metadata collections, and finding interesting patterns. We showed how this type of exploration can support tasks such as the distribution of goals and other salient events *vs* type of match and match phase, and the variation of loudness patterns with respect to other data dimensions.

Several observations can be made about our results concerning the second research question. First and foremost, exploring multimedia collections is an extremely wide topic. Such collections are of numerous, and different, types, involving many attributes of various kinds (static image features, annotations, metadata, temporal dynamics in audio and/or video data, to mention just a few). Such attributes come in a wide set of formats, and their actual values, ranges, precision, and – most importantly – semantics highly differ as function of the actual extraction techniques used. As such, the added-value and insights obtained by visually exploring such data are extremely dependent on the actual multidimensional data extraction process. The insights obtained by our proposed visualization are, in turn, necessarily constrained to the quality of the available data-extraction tools. Better and more elaborate insights can be clearly obtained once one avails of better data-extraction tools. To the present moment, however, such tools are not easily available publicly. Secondly, the questions and interests of users may widely vary depending on who they are, but also on the semantics of the concrete dataset under exploration. As such, it is arguably not possible to design a *generically* optimal visual exploration technique or tool for such data. We argue that our solutions provide added-value for the tasks they were aimed at (organizing and browsing a static multimedia collection, and browsing and comparing videos from a soccer video collection). For other tasks and/or types of multimedia datasets, completely different visual metaphors and/or designs will likely be more suitable.

The use of projections in exploring both static multimedia collections (Chapter 3) and video collections (Chapter 6) jointly reveal another important insight. On the one hand, when augmented by suitable interaction and visual presentation techniques, projections can be very effective instruments for high-level exploration tasks, such as collection organization and semantic interpretation of groups and outliers (Chapter 3). Conversely, when one limits oneself to using static projections color-coded by the value of a dimension and annotated by biplot axes, their ease-of-use and types of insights these can provide, are limited to lower-level technical tasks such as finding outliers and correlated dimensions and dimension-ranges (Chapter 6). This points to the crucial need of augmenting projections by suitable *explanatory* annotations. Currently, such annotations are still relatively low-level – they cannot, for example, produce a view in which we immediately see what is the meaning of groups of close points, or why is an outlier far away from the surrounding points. Such limitations are important for all contexts where multidimensional projections are used, beyond the scope of our thesis.

7.1 FUTURE WORK

The results presented in this thesis through methods and applications open several directions for future research. In line with the discussion listed above, we identify the following high-potential work directions:

Projection computation: To make projections more effective data-exploration instruments, we need to improve the data transformation and interpretation pipeline they propose at several points. A first point here includes the design of automatic or semi-automatic ways to set the many parameters of a projection can be designed, based on user specification of the minimization of given projection errors over given subsets of observations. This would effectively close the feedback loop formed by interpreting projection errors and leading to the optimization of the projection to reduce such errors. Separately, this would make the usage of projections far easier for non-specialists. For the scope of tuning projections by control points (Chapter 3), a detailed study of the desirable quantity of control points and sizes of their k-nearest neighborhoods could lead to the development of more accurate projection control system.

Projection quality: As outlined above, specific tasks are affected by projection errors in specific ways. The currently known error metrics and their related visualizations cover only a small subset of generic cases. These can be refined to include more specific error patterns, and correspondingly customized visualizations for these. Examples include studying neighborhood-preservation and group-preservation errors, studying the effect of measurement uncertainty on the patterns created in the projection, and studying how a projection's quality is dynamically affected by the interactive manipulation of control points mentioned above.

Projection explanation: Arguably one of the main points of improvement of effectiveness of projections in practice relates to making them more self-explaining. Putting it simply, we need ways to communicate to non-specialists *why* projected points are similar, much like maps and infographics include annotations that explain their elements in simple terms. This could be achieved by detecting the visual patterns appearing in a projection, analyzing the key elements that characterize such pattern, and designing compact and illustrative visual encodings to depict these elements. To make such techniques scalable to dense projections containing hundreds of thousands of observations, image-based methods are techniques of choice. Extending such explanatory techniques to 3D projections is a separate important point, which we feel to be critical for the acceptance of 3D projections in practice.

Multimedia data exploration: Considering that multimedia has a huge reach, and that current tools and techniques for exploring large multimedia collections are still quite limited, designing better exploration tools has a large potential. In detail, we see current tools as being either very insightful, but focusing on expert users; or very easy to use, but covering only simple queries. Projections offer, in our view, the best direction towards the creation of 'data landscapes' able to visualize how multimedia data are self-organized into groups, patterns, and outliers. Projection-based techniques can be extended by considering additional attributes extracted from multimedia data (including categorical and relational ones). Additionally, projection-based techniques can be applied in different multimedia-related contexts. For instance, in video surveillance videos, projection techniques could be used to identify and show in real time anomalous behavior occurring in tens or hundreds of live streams, or to generate dynamic summarizations of anomalous events over given time spans. For the sports domain, our propose techniques could be extended to handle live broadcasts via multimedia streaming. Such applications could detect and classify events, perform tracking and identification of players, create high-level overviews of the extracted data, and highlight game patterns in such data.

Other time-dependent data: While our main applications of multidimensional projections to dynamic data has been in the context of multimedia, many other types of multidimensional dynamic data exist. These include, but are not limited to, vehicle flows captured by route- or air-traffic-control systems; dynamics of intraday stocks on the stock market; and generally any time-dependent phenomenon that is characterized by a large set of time-varying signals. Studying how multidimensional projections can be extended to give insight into such dynamic data to non-specialist users is probably the largest, but also most important, challenge related to the work presented in this thesis.

BIBLIOGRAPHY

[1] H. Abdi and D. Valentin. *Encyclopedia of Measurement and Statistics*. Thousand Oaks, 2007. Ch. *Multiple Correspondence Analysis*.

[2] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park. Fast algorithms for projected clustering. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 61–72, 1999.

[3] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, volume 22, pages 207–216, 1993.

[4] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 94–105, 1998.

[5] M. Ajmal, M. Ashraf, M. Shakir, Y. Abbas, and F. Shah. Video summarization: Techniques and classification. In L. Bolc, R. Tadeusiewicz, L. Chmielewski, and K. Wojciechowski, editors, *Computer Vision and Graphics*, volume 7594 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2012.

[6] A. Anand, L.Wilkinson, and T. Dang. Visual pattern discovery using random projections. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*, pages 43–52, 2012.

[7] M. Ankerst. Visual data mining with pixel-oriented visualization techniques. In *Proceedings of the ACM SIGKDD Workshop on Visual Data Mining*, 2001.

[8] M. Ankerst, D. A. Keim, and H.-P. Kriegel. Circle segments : A technique for visually exploring large multidimensional data sets. In *Hot Topics paper presented at presented at IEEE Visualization 96, San Francisco*, 1996.

[9] D. Arumugam, M. Sibley, J. Griffin, D. Stancil, and D. Ricketts. An active position sensing tag for sports visualization in american football. In *RFID (RFID), 2013 IEEE International Conference on*, pages 96–103, 2013.

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

[11] M. E. Ayadi, M. S. Kamel, and F. Karray. Survey on speech emotion recognition: Features, classification schemes, and databases. *Pattern Recognition*, 44(3):572–587, 2011.

[12] V. Barnett. The ordering of multivariate data. *Journal of the Royal Statistical Society. Series A (General)*, 139(3):318–355, 1976.

[13] 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.

[14] J. Beddow. Shape coding of multidimensional data on a microcomputer display. In *Proceedings of the 1st Conference on Visualization '90*, pages 238–246. IEEE Computer Society Press, 1990.

[15] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[16] R. Bellman. *Adaptive Control Processes*. Princeton University Press, 1961.

[17] T. Bergmann, S. Bunk, J. Eschrig, C. Hentschel, M. Knuth, H. Sack, and R. Schüler. Generating a linked soccer dataset. In *Proceedings of the 9th International Conference on Semantic Systems (I-SEMANTICS '13)*, pages 146–149, 2013.

[18] M. W. Berry and M. Castellanos. *Survey of Text Mining II*. Springer, 2007.

[19] J. Bertin. *Semiology of graphics: diagrams, networks, maps*. University of Wisconsin Press, 1983.

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

[21] L. S. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington, and R. C. Whaley. An updated set of basic linear algebra subprograms (BLAS). *ACM Transactions on Mathematical Software*, 28(2):135–151, June 2002.

[22] R. Borgo, M. Chen, B. Daubney, E. Grundy, G. Heidemann, B. Haeferlin, M. Haeferlin, H. Leitte, D. Weiskopf, and X. Xie. State of the art report on video-based graphics and video visualization. *Computer Graphics Forum*, 31(8):2450–2477, 2012.

[23] H. Bornman and S. Von Solms. Hypermedia, multimedia and hypertext: definitions and overview. *The Electronic Library*, 11(4/5):259–268, 1993.

[24] U. Brandes and C. Pich. Eigensolver methods for progressive multidimensional scaling of large data. In M. Kaufmann and D. Wagner, editors, *Graph Drawing*, volume 4372 of *Lecture Notes in Computer Science*, pages 42–53. Springer Berlin Heidelberg, 2007.

[25] M. L. Braun, J. Schaback, M. L. Jugel, and N. Oury. jBlas: Linear algebra for Java, 2011. URL `http://www.jblas.org/`.

[26] C. Brewer and M. Harrower. ColorBrewer, 2014. `www.colorbrewer.org`.

[27] B. Broeksema. *The Decision Exploration Lab: Supporting the business analyst in understanding automated decisions*. PhD thesis, Johann Bernoulli Institute for Mathematics and Computer Science, University of Groningen, The Netherlands, July 2014. URL `http://www.cs.rug.nl/~alext/PAPERS/PhD/broeksema14.pdf`. Promotors: A. C. Telea, G. Melancon.

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

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

[30] M. Bruls, K. Huizing, and J. J. van Wijk. Squarified treemaps. In *Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization*, pages 33–42, Berlin, 2000. Springer.

[31] C. J. Burges. Dimension reduction: A guide tour. *Foundations and Trends in Machine Learning*, 2(4):275–365, 2009.

[32] 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 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 83–90, 2010.

[33] CCSL/IME-USP. Datsaet used in the paper "A study of the relationships between source code metrics and attractiveness in free software projects" (Meirelles *et al.*, 2012), 2013. `http://ccsl.ime.usp.br/mangue/data`. Last access: 15/07/2013.

[34] M. Chalmers. A linear iteration time layout algorithm for visualising high-dimensional data. In *Visualization'96. Proceedings*, pages 127–131. IEEE, 1996.

[35] J. M. Chambers. *Graphical Methods for Data Analysis*, volume xiv of *Wadsworth Statistics/Probability Series*. Duxbury Press, 1983.

[36] W. Y. Chan. A survey on multivariate data visualization, June 2006. Technical Report HKUST/06/2006, Department of Computer Science and Engineering, Hong Kong University of Science and Technology, available at `http://people.stat.sc.edu/hansont/stat730/multivis-report-winnie.pdf`.

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

[38] T. Chen, A. Lu, and S. Hu. Visual storylines: Semantic visualization of movie sequence. *Computers & Graphics*, 2012.

[39] Y. Chen, L. Wang, M. Dong, and J. Hua. Exemplar-based visualization of large document corpus. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1161–1168, 2009.

[40] C.-H. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '99)*, pages 84–93, New York, NY, USA, 1999.

[41] Y. Cheng and G. Church. Biclustering of expression data. *Proceedings of International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 8:93–103, 2000.

[42] H. Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, 68:361–368, 1973.

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

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

[45] C. D. Correa and K.-L. Ma. Dynamic video narratives. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, pages 88:1–88:9. ACM, 2010.

[46] 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.

[47] A. Criminisi and J. Shotton. *Decision Forests for Computer Vision and Medical Image Analysis*. Springer, 2013.

[48] W. Cui, Y. Wu, S. Liu, F. Wei, M. Zhou, and H. Qu. Context-preserving, dynamic word cloud visualization. *Computer Graphics and Applications, IEEE*, 30(6):42–53, Nov 2010.

[49] R. R. O. da Silva, P. Rauber, R. Martins, R. Minghim, and A. Telea. Attribute-based visual explanation of multidimensional projections. In *Proc. EuroVA*, pages 137–143. Eurographics, 2015.

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

[51] J. Daniels II, E. Anderson, L. Nonato, and C. Silva. Interactive vector field feature identification. *Visualization and Computer Graphics (TVCG), IEEE Transactions on*, 16(6):1560–1568, 2010.

[52] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):357–366, 1980.

[53] M. J. J. de Hoon, S. Imoto, J. Nolan, and S. Miyano. Open source clustering software. *Bioinformatics*, 19:1453–1454, 2004. `http://bonsai.hgc.jp/$\sim$mdehoon/software/cluster`.

[54] V. de Silva and J. Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Stanford Univ., 2004. `window.stanford.edu/courses/cs468-05-winter/Papers/Landmarks/Silva_landmarks5.pdf`. Last access: 15/07/2013.

[55] I. Demir and R. Westermann. Progressive high-quality response surfaces for visually guided sensitivity analysis. *Computer Graphics Forum*, 32(3pt1): 21–30, 2013.

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

[57] S. Diehl. *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Springer, Berlin, 2010.

[58] S. Diehl and A. C. Telea. Multivariate networks in software engineering. In A. K. *et al.*, editor, *Multivariate Network Visualization*, volume 8830, pages 13–36. Springer LNCS, 2013.

[59] T. D'Orazio and M. Leo. A review of vision-based systems for soccer video analysis. *Pattern Recognition*, 43(8):2911 – 2926, 2010.

[60] S. dos Santos and K. Brodlie. Gaining understanding of multivariate and multidimensional data through visualization. *Computers & Graphics*, 28(3): 311 – 325, 2004.

[61] E. dos Santos Amorim, E. V. Brazil, J. Daniels, P. Joia3, L. G. Nonato, and M. C. Sousa. iLAMP: Exploring high-dimensional spacing through backward multidimensional projection. In *Visual Analytics Science and Technology (VAST), IEEE Conference on*, pages 53–62, 2012.

[62] J. Dykes, A. M. MacEachren, and M.-J. Kraak. *Exploring Geovisualization*. Elsevier, 2005.

[63] N. Elmqvist, J. Stasko, and P. Tsigas. Datameadow: A visual canvas for analysis of large-scale multivariate data. In *Visual Analytics Science and Technology (VAST 2007), IEEE Symposium on*, pages 187–194, Oct 2007.

[64] N. Elmqvist, P. Dragicevic, and J. Fekete. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *Visualization and Computer Graphics (TVCG), IEEE Transactions on*, 14(6):1539–1148, 2008.

[65] A. Endert, P. Flaux, and C. North. Semantic interaction for visual text analytics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (ACM CHI '12)*, pages 324–333, 2012.

[66] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. 2nd International Conference On Knowledge Discovery And Data Mining*, pages 226–231, 1996.

[67] S. Fabrikant. *Spatial metaphors for browsing large data archives*. PhD thesis, PhD thesis, Univ. of Colorado Boulder, 2000.

[68] S. G. Fadel, F. M. Fatore, F. S. Duarte, and F. V. Paulovich. LoCH: a neighborhood-based multidimensional projection technique for high-dimensional sparse spaces. *Neurocomputing*, 150, Part B(0):546 – 556, 2015.

[69] C. Faloutsos and K. Lin. FastMap: A fast algorithm for indexing, datamining and visualization of traditional and multimedia databases. In *Proc. 1995 ACM SIGMOD International Conference on Management of Data*, pages 163–174, 1995.

[70] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition, 2003 IEEE Computer Society Conference on*, volume 2, pages 264–271, 2003.

[71] A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

[72] Freefoto. Free stock photo collection, 2013. www.freefoto.com. Last access: 15/07/2013.

[73] Y. Frishman and A. Tal. Multi-level graph layout on the gpu. *Visualization and Computer Graphics (TVCG), IEEE Transactions on*, 13(6):1310–1319, 2007.

[74] I. Fuinaga and D. McEnnis. On-demand metadata extraction network (OMEN). In *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '06)*, pages 346–346, 2006.

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

[76] H. Goeau, J. Thievre, M.-L. Viaud, and D. Pellerin. Interactive visualization tool with graphic table of video contents. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 807–810, 2007.

[77] B. Goethals. Survey on frequent pattern mining. In *Univ. of Helsinki*, 2003.

[78] J. Gower and G. Dijksterhuis. *Procrustes Problems*. Oxford University Press, 2004.

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

[80] J. C. Gower and D. J. Hand. *Biplots*, volume 54. CRC Press, 1995.

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

[82] J. F. Hair, W. C. Black, B. J. Babin, R. E. Anderson, and R. L. Tatham. *Multivariate data analysis*, volume 6. Pearson Prentice Hall, 2006.

[83] J. Han, M. Kamber, and J. Pei. *Data mining: Concepts and Techniques*. Elsevier, 2011.

[84] D. Hand, H. Mannila, and P. S. P. *Principles of Data Mining*. MIT Press, 2001.

[85] A. Hanjalic. *Content-Based Analysis of Digital Video*. Springer Science, Kluwer Academic Publishers, 2004. 193 pages.

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

[87] HCI Laboratory, Univ. Maryland. Human computer interaction example datasets, 2013. http://www.cs.umd.edu/hcil/hce/examples/application_examples.html.

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

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

[90] P. Hoffman, G. Grinstein, K. Marx, I. Grosse, and E. Stanley. DNA visual and analytic data mining. In *Proceedings of the 8th Conference on Visualization '97 (VIS '97)*, pages 437–441, 1997.

[91] A. Hotho, A. Nurnberger, and G. Paa. A brief survey of text mining. *Journal of Computational Linguistics and Language Technology*, 20(1):19–62, 2005.

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

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

[94] C. Hurter, O. Ersoy, and A. Telea. Smooth bundling of large streaming and sequence graphs. In *Visualization Symposium (PacificVis), 2013 IEEE Pacific*, pages 41–48. IEEE Press, 2013.

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

[96] C. Hurter, A. Taylor, S. Carpendale, and A. Telea. Color tunneling: Interactive exploration and selection in volumetric datasets. In *Proc. IEEE PacificVis*, pages 327–335, 2014.

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

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

[99] A. Inselberg and B. Dimsdale. Parallel coordinates: a tool for visualizing multi-dimensional geometry. In *Proc. IEEE Visualization*, pages 361–378, 1990.

[100] IPTV-News. World cup final records for tv broadcasters in 2014, 2014. `http://www.iptv-news.com/2014/07/world-cup-final-breaksrecords-worldwide-for-tv-broadcasters`.

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

[102] A. K. Jain, M. Murthy, and P. Flynn. Data clustering: A review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

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

[104] P. Joia, F. V. Paulovich, D. Coimbra, J. Cuminato, and L. Nonato. Lamp Video Demonstration, 2011. `http://ieeexplore.ieee.org/ielx5/2945/6064926/6065024/ttg2011122563.zip?tp=&arnumber=6065024`.

[105] I. T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[106] F. Jourdan and G. Melançon. Multiscale hybrid MDS. In *Information Visualisation*, pages 388–393, 2004.

[107] C. Judd, G. McClelland, and C. Ryan. *Data Analysis: A Model Comparison Approach, Second Edition*. Taylor & Francis, 2011.

[108] E. Kandogan. Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. In *Proceedings of the IEEE Information Visualization Symposium, Late Breaking Hot Topics*, pages 9–12, 2000.

[109] E. Kandogan. Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 107–116, 2001.

[110] D. Keim. Information visualization and visual data mining. *Visualization and Computer Graphics, IEEE Transactions on*, 8(1):1–8, 2002.

[111] D. Keim, J. Hohlhammer, G. Ellis, and F. Mansmann, editors. *Mastering the information age – solving problems with visual analytics*. Eurographics, 2010.

[112] D. A. Keim. Visual techniques for exploring databases. In *Proc. 3rd Intl. Conf. on Knowledge Discovery and Data Mining (tutorial program)*, 1997. available at `http://www.informatik.uni-halle.de/~keim/PS/KDD97.pdf`.

[113] D. A. Keim and H.-P. Kriegel. Visdb: Database exploration using multidimensional visualization. *IEEE Computer Graphics and Applications*, 14: 40–49, 1994.

[114] D. A. Keim and H.-P. Kriegel. Visualization techniques for mining large databases: A comparison. *Knowledge and Data Engineering, IEEE Transactions on*, 8(6):923–938, 1996.

[115] A. Khacharem, B. Zoudji, S. Kalyuga, and H. Ripoll. Developing tactical skills through the use of static and dynamic soccer visualizations:an expert-nonexpert differences investigation. *Journal of Applied Sport Psychology*, 25(3):326–340, 2013.

[116] T. Kohonen. *Self-Organization and Associative Memory (3rd edition)*. Berlin: Springer-Verlag, 1989.

[117] V. Kolar. Vortex identification: New requirements and limitations. *International Journal of Heat and Fluid Flow*, 28:638–652, 2007.

[118] M. H. Kolekar. Bayesian belief network based broadcast sports video indexing. *Multimedia Tools and Applications*, 54(1):27–54, 2011.

[119] Y. Koren, L. Carmel, and D. Harel. ACE: A fast multiscale eigenvectors computation for drawing huge graphs. In *Information Visualization (INFOVIS 2002). IEEE Symposium on*, pages 137–144, 2002.

[120] KPMG International Cooperative. Going beyond the data: Achieving actionable insights with data and analytics, 2014. `https://www.kpmg.com/Global/en/IssuesAndInsights/ArticlesPublications/Documents/going-beyond-data-and-analytics-v4.pdf`.

[121] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, editor, *Advances in Neural Information Processing Systems*, pages 1097–1105. Curran, Inc., 2012.

[122] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:115–129, 1964.

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

[124] F. Lederman. Parvis parallel coordinates visualisation, 2012. http://home.subnet.at/flo/mv/parvis/.

[125] 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.

[126] R. Leonardi, P. Migliorati, and M. Prandini. Semantic indexing of soccer audio-visual sequences: a multimodal approach based on controlled markov chains. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(5):634–643, 2004.

[127] 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.

[128] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[129] H. Levkowitz. Color icons: Merging color and texture perception for integrated visualization of multiple parameters. In *Proceedings of the 2nd conference on Visualization'91*, pages 164–170, 1991.

[130] J. Lewis, L. van der Maaten, and V. de Sa. A behavioral investigation of dimensionality reduction. In *Proceedings of 34th Conference of the Cognitive Science Society (CogSci)*, pages 671–676, 2012.

[131] J. Li and J. Z. Wang. Automatic Linguistic Indexing of Pictures by a Statistical Modeling Approach. *Pattern Analysis and Machine Intelligence (TPAMI), IEEE Transactions*, 25:1075–1088, 2003.

[132] L.-J. Li and L. Fei-Fei. Optimol: Automatic online picture collection via incremental model learning. *International Journal of Computer Vision*, 88(2): 147–168, 2010.

[133] B. Lichtenbelt, R. Crane, and S. Naqvi. *Introduction to Volume Rendering*. Prentice Hall - Hewlett-Packard Professional Books, 1998.

[134] S. Liu, Y. Wu, E. Wei, M. Liu, and Y. Liu. Storyflow: Tracking the evolution of stories. *Visualization and Computer Graphics (TVCG), IEEE Transactions on*, 19(12):2436–2445, 2013.

[135] S. Liu, D. Maljovec, B. Wang, P.-T. Bremer, and V. Pascucci. Visualizing High-Dimensional Data: Advances in the Past Decade. In R. Borgo, F. Ganovelli, and I. Viola, editors, *Eurographics Conference on Visualization (EuroVis) - STARs*. The Eurographics Association, 2015.

[136] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International*, volume 2, pages 1150–1157. IEEE Press, 1999.

[137] P. Lucey, D. Oliver, P. Carr, J. Roth, and I. Matthews. Assessing team strategy using spatiotemporal data. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*, pages 1366–1374, 2013.

[138] H. Luo, J. Fan, J. Yang, W. Ribarsky, and S. Satoh. Exploring large-scale video news via interactive visualization. In *Visual Analytics Science And Technology (VAST 2006), IEEE Symposium On*, pages 75–82, 2006.

[139] H. Luo, J. Fan, J. Yang, W. Ribarsky, and S. Satoh. Analyzing large-scale news video databases to support knowledge visualization and intuitive retrieval. In *Visual Analytics Science and Technology (VAST 2007), IEEE Symposium on*, pages 107–114, 2007.

[140] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. L. Cam and J. Neyman, editors, *Proceedings Of The Fifth Berkeley Symposium On Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

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

[142] 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 *Software Engineering (SBES), 2010 Brazilian Symposium on*, pages 11–20, 2012.

[143] Mercer Marsh & McLellan. Turning data into actionable insights, 2014. `http://mthink.mercer.com/turning-data-into-actionable-insights`.

[144] A. Morrison, G. Ross, and M. Chalmers. A hybrid layout algorithm for sub-quadratic multidimensional scaling. In *IEEE Information Visualization*, pages 152–158, 2002.

[145] D. Mount and S. Arya. Approximate nearest neighbor search software, 2011. `www.cs.umd.edu/~mount/ANN`. Last access: 15/07/2013.

[146] T. Mühlbacher, H. Piringer, S. Gratzl, M. Sedlmair, and M. Streit. Opening the black box: Strategies for increased used involvement in existing algorithmic implementations. *Visualization and Computer Graphics (TVCG), IEEE Transactions on*, 20(12):1643–1652, 2014.

[147] T. Munzner. *Visualization Analysis and Design*. CRC Press, 2015.

[148] T. Munzner, F. Guimbretiére, S. Tasiran, L. Zhang, and Y. Zhou. TreeJuxtaposer: scalable tree comparison using focus+context with guaranteed visibility. *ACM Transactions on Graphics (TOG)*, 22(3):453–462, 2003.

[149] G. B. Newby. Empirical study of a 3D visualization for information retrieval tasks. *Journal of Intelligent Information Systems*, 18(1):31–53, 2002.

[150] N. Nguyen and A. Yoshitaka. Soccer video summarization based on cinematography and motion analysis. In *Multimedia Signal Processing (MMSP), IEEE 16th International Workshop on*, pages 1–6, 2014.

[151] M. Norman and D. Whalen. IEEE Vis'08 contest data, 2013. *scivisconttest.ieeevis.org/2008*.

[152] S. Oeltze, H. Doleisch, H. Hauser, P. Muigg, and B. Preim. Interactive visual analysis of perfusion data. *Visualization and Computer Graphics (TVCG), IEEE Transactions on*, 13(6):1392–1399, 2007.

[153] 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.

[154] P. Oskouie, S. Alipour, and A.-M. Eftekhari-Moghadam. Multimodal feature extraction and fusion for semantic mining of soccer video: a survey. *Artificial Intelligence Review*, 42(2):173–210, 2014.

[155] Ovum, Inc. World cup broadcasters target 6 billion screens, 2014. `http://www.ovum.com/press_releases/world-cup-broadcasters-target-6-billion-screens-and-4-7-billion-connected-devi`

[156] C. E. P. Zikopoulos. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill, 2011.

[157] M. Page and A. V. Moere. Towards classifying visualization in team sports. In *Proc. Intl. Conf. on Computer Graphics, Imaging and Visualisation*, pages 24–29, 2006.

[158] 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.

[159] J. G. Paiva, L. Florian, H. Pedrini, G. P. Telles, and R. Minghim. Improved similarity trees and their application to visual data classification. *Visualization and Computer Graphics (TVCG), IEEE Transactions on*, 17(12):2459–2468, 2011. urlhttp://infoserver.lcad.icmc.usp.br/infovis2/Tools.

[160] F. Pan, G. Cong, A. K. H. Tung, J. Yang, and M. J. Zaki. Carpenter: Finding closed patterns in long biological datasets. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*, KDD 03, pages 637–642. ACM, 2003.

[161] T. Pang-Ning, M. Steinbach, V. Kumar, et al. Introduction to data mining. In *Library of Congress*, page 74, 2006.

[162] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Database Theory-ICDT99*, pages 398–416. Springer, 1999.

[163] F. Paulovich, L. Nonato, R. Minghim, and H. Levkowitz. Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. *Visualization and Computer Graphics, IEEE Transactions on*, 14(3):564–575, 2008.

[164] F. Paulovich, C. Silva, and L. Nonato. Two-phase mapping for projecting massive data sets. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1281–1290, 2010.

[165] F. V. Paulovich and R. Minghim. HiPP: A novel hierarchical point placement strategy and its application to the exploration of document collections. *IEEE Trans. Visual. Comp. Graph.*, 14(6):1229–1236, 2008.

[166] F. V. Paulovich, L. G. Nonato, and R. Minghim. Visual mapping of text collections through a fast high precision projection technique. In *Information Visualization (IV), Tenth International Conference on*, pages 282–290, 2006.

[167] F. V. Paulovich, D. Eler, J. Poco, C. Botha, R. Minghim, and L. Nonato. Piece wise laplacian-based projection for interactive data exploration and organization. *Computer Graphics Forum*, 30(3):1091–1100, 2011.

[168] F. V. Paulovich, C. T. Silva, and L. G. Nonato. User-centered multidimensional projection techniques. *Computing in Science & Eng*, 14(4):74–81, 2012.

[169] F. V. Paulovich, F. M. B. Toledo, G. P. Telles, R. Minghim, and L. G. Nonato. Semantic wordification of document collections. *Computer Graphics Forum*, 31(3pt3):1145–1153, 2012.

[170] J. Pei, J. Han, R. Mao, et al. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD workshop on research issues in data mining and knowledge discovery*, volume 4, 2000.

[171] E. Pekalska, D. de Ridder, R. Duin, and M. Kraaijveld. A new method of generalizing Sammon mapping with application to algorithm speed-up. In *Proc. ASCI*, pages 221–228, 1999.

[172] C. Perin, R. Vuillemot, and J.-D. Fekete. Soccerstories: A kick-off for visual soccer analysis. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2506–2515, Dec 2013.

[173] C. Perin, R. Vuillemot, and J.-D. Fekete. À table!: Improving temporal navigation in soccer ranking tables. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, pages 887–896, 2014.

[174] R. M. Pickett and G. G. Grinstein. Iconographic displays for visualizing multidimensional data. In *Proceedings of the 1988 IEEE Conference on Systems, Man, and Cybernetics*, volume 1, pages 514–519, 1988.

[175] H. Pileggi, C. Stolper, J. Boyle, and J. Stasko. Snapshot: Visualization to propel ice hockey analytics. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2819–2828, 2012.

[176] H. Piringer, R. Kosara, and H. Hauser. Interactive f+c visualization with linked 2D/3D scatterplots. In *Coordinated and Multiple Views in Exploratory Visualization, Proceedings of Second International Conference on*, pages 49–60, 2004.

[177] H. Piringer, W. Berger, and J. Krasser. Hypermoval: Interactive visual validation of regression models for real-time simulation. *Computer Graphics Forum*, 29(3):983–992, 2010.

[178] 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.

[179] T. Polk, J. Yang, Y. Hu, and Y. Zhao. Tennivis: Visualization for tennis match analysis. *Visualization and Computer Graphics (TVCG), IEEE Transactions on*, 20(12):2339–2348, 2014.

[180] A. J. Pretorius and J. J. V. Wijk. Visual analysis of multivariate state transition graphs. *Visualization and Computer Graphics (TVCG), IEEE Transactions on*, 12(5):685–692, 2006.

[181] R. Prokop and A. Reeves. A survey of moment-based techniques for unoccluded object representation and recognition. *Graphical Models and Image Processing*, 54(5):438–460, 1992.

[182] Y. Qi, A. Hauptmann, and T. Liu. Supervised classification for video shot segmentation. In *Multimedia and Expo (ICME '03) ,Proceedings of International Conference on*, volume 2, pages 689–692, 2003.

[183] C. Ramussen and C. Williams. Gaussian processes for machine learning (adaptive computation and machine learning), 2006.

[184] 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 ACM Conference on Human Factors in Computing Systems (CHI)*, pages 318–322, 1994.

[185] A. Raventos, R. Quijada, L. Torres, F. Tarre's, E. Carasusa'n, and D. Giribet. The importance of audio descriptors in automatic soccer highlights generation. In *Multi-Conference on Systems, Signals & Devices (SSD), 11th International*, pages 1–6, 2014.

[186] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[187] M. Rumpf and A. Telea. A continuous skeletonization method based on level sets. In *Proceedings of the symposium on Data Visualisation 2002*, pages 151–159. Eurographics Association, 2002.

[188] A. Rusu, D. Stoica, E. Burns, B. Hample, K. McGarry, and R. Russell. Dynamic visualizations for soccer statistical analysis. In *Information Visualisation (IV), 2010 14th International Conference*, pages 207–212, 2010.

[189] G. Salton. *Introduction to modern information retrieval.* McGraw, 1986.

[190] G. Salton. Developments in automatic text retrieval. *Science*, 253:974–980, 1991.

[191] H. Samet. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann, 1 edition, 2006.

[192] H. Sanftmann. *3D visualization of multivariate data*. PhD thesis, Univ. Stuttgart, Germany, 2014. `http://elib.uni-stuttgart.de/opus/volltexte/2012/7807`.

[193] H. Sanftmann and D. Weiskopf. Illuminated 3D scatterplots. *Computer Graphics Forum*, 28(3):751–758, 2009.

[194] H. Sanftmann and D. Weiskopf. 3D scatterplot navigation. *Visualization and Computer Graphics (TVCG), IEEE Transactions on*, 18(11):1969–1978, 2012.

[195] S. Schaefer, T. McPhail, and J. Warren. Image deformation using moving least squares. *ACM Transactions on Graphics (TOG)*, 25(3):533–540, 2006.

[196] K. Schoeffmann, F. Hopfgartner, O. Marques, L. Boeszoermenyi, and J. M. Jose. Video browsing interfaces and applications: a review. *Journal of Photonics for Energy*, pages 1804–1835, 2010.

[197] T. Schreck, T. von Landesberger, and S. Bremm. Techniques for precision-based visual analysis of projected data. *Information Visualization*, 9(3):181–193, 2010.

[198] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3-D Graphics, 4rd edition*. Kitware, Inc., Clifton Park, NY:, 2006.

[199] G. Schwarzer, W. Vach, and M. Schumacher. On the misuses of artificial neural networks for prognostic and diagnostic classification in oncology. *Statistics in Medicine*, 19:541–561, 2000.

[200] M. Sedlmair, A. Tatu, T. Munzner, and M. Tory. A taxonomy of visual cluster separation factors. *CGF*, 31(3):1335–1344, 2012.

[201] M. Sedlmair, T. Munzer, and M. Tory. Empirical guidance on scatterplot and dimension reduction technique choices. *Visualization and Computer Graphics (TVCG), IEEE Transactions on*, 19(12):2634–2643, 2013.

[202] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic imaging*, 13(1): 146–165, 2004.

[203] J. R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22(1–3):21–74, 2002.

[204] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Visual Languages, Proceedings of IEEE Symposium on*, pages 336–343, 1996.

[205] V. D. Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems*, volume 15, pages 705–712. MIT Press, 2002.

[206] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *Pattern Analysis and Machine Intelligence (TPAMI), IEEE Transactions on*, 22(12):1349–1380, 2000.

[207] A. Smola and B. Schlkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[208] C. G. M. Snoek and M. Worring. Multimodal video indexing: A review of the state-of-the-art. *Multimedia tools and applications*, 25(1):5–35, 2005.

[209] R. O. Stehling, M. A. Nascimento, and A. X. Falcão. A compact and efficient image retrieval approach based on border/interior pixel classification. In *Proceedings of the eleventh international conference on Information and knowledge management (CIKM)*, pages 102–109. ACM, 2002.

[210] F. Sulser, I. Giangreco, and H. Schuldt. Crowd-based semantic event detection and video annotation for sports videos. In *Proceedings of the 2014 International ACM Workshop on Crowdsourcing for Multimedia (CrowdMM)*, pages 63–68, 2014.

[211] P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[212] M. Tavanti and M. Lind. 2D vs 3D, implications on spatial memory. In *Information Visualization (INFOVIS 2001), IEEE Symposium on*, pages 139–145, 2001.

[213] M. Tavassolipour, M. Karimian, and S. Kasaei. Event detection and summarization in soccer videos using bayesian network and copula. *Circuits and Systems for Video Technology, IEEE Transactions on*, 24(2):291–304, 2014.

[214] 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.

[215] A. Telea. Combining extended table lens and treemap techniques for visualizing tabular data. In *Proceedings of the Eighth Joint Eurographics/IEEE VGTC conference on Visualization*, pages 51–58. Eurographics Association, IEEE Press, 2006.

[216] A. Telea and O. Ersoy. Image-based edge bundles: Simplified visualization of large graphs. *Computer Graphics Forum*, 29(3):843–852, 2010.

[217] A. C. Telea. *Data visualization: principles and practice*. CRC Press, 2014. 2nd edition.

[218] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[219] J. A. Thomas and K. A. Cook, editors. *Illuminating the Path: Research and Development Agenda for Visual Analytics*. IEEE Press, Los Alamitos, CA, 2005.

[220] W. Torgerson. Multidimensional scaling of similarity. *Psychometrika*, 30(4): 379–393, 1965.

[221] B. T. Truong and S. Venkatesh. Video abstraction: A systematic review and classification. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 3(1), 2007.

[222] E. R. Tufte. *The Visual Display of Quantitative Information (2nd edition)*. Graphics Press, 2001.

[223] 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.

[224] C. Turkay, P. Filzmoser, and H. Hauser. Brushing dimensions-a dual visual analysis model for high-dimensional data. *Visualization and Computer Graphics (TVCG), IEEE Transactions on*, 17(12):2591–2599, 2011.

[225] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2431–2456, 2008.

[226] L. van der Maaten and G. Hinton. Visualizing non-metric similarities in multiple maps. *Machine Learning*, 87(1):33–35, 2012.

[227] 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.

[228] J. J. Van Wijk and H. van de Wetering. Cushion treemaps: Visualization of hierarchical information. In *Information Visualization (Info Vis' 99), Proceedings of IEEE Symposium on*, pages 73–81, 1999.

[229] V. N. Vapnik and V. Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[230] 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.

[231] W. Wang and J. Yang. Mining high-dimensional data. In *Data Mining and Knowledge Discovery Handbook*, pages 803–808. Springer, 2010.

[232] Y. Wang, Z. Liu, and J.-C. Huang. Multimedia content analysis-using both audio and visual clues. *Signal Processing Magazine, IEEE*, 17(6):12–36, 2000.

[233] M. O. Ward, G. Grinstein, and D. Keim. *Interactive data visualization: foundations, techniques and applications*. CRC Press, 2010.

[234] M. Wattenberg. Visual exploration of multivariate graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 811–819, 2006.

[235] X. Wei, L. Sha, P. Lucey, S. Morgan, and S. Sridharan. Large-scale analysis of formations in soccer. In *Digital Image Computing: Techniques and Applications (DICTA), IEEE International Conference on*, pages 1–8, 2013.

[236] S. Westerman and T. Cribbin. Mapping semantic information in virtual space: dimensions, variance and individual differences. *International Journal of Human-Computer Studies (IJHCS)*, 53(5):765–787, 2000.

[237] 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 (IJHCS)*, 62(6):713–736, 2005.

[238] D. Whalen and M. L. Norman. Competition data set and description. In *2008 IEEE Visualization Design Contest*. http://vis.computer.org/VisWeek2008/vis/contests.html, 2008.

[239] L. Wilkinson, A. Anand, and R. L. Grossman. Graph-theoretic scagnostics. In *Proc. IEEE InfoVis*, volume 5, page 21, 2005.

[240] P. C. Wong. Guest editor's introduction: Visual data mining. *IEEE Computer Graphics and Applications*, 19(5):20–21, 1999.

[241] M. Worring, C. Snoek, O. de Rooij, G. Nguyen, and A. Smeulders. The mediamill semantic video search engine. In *Acoustics, Speech and Signal Processing (ICASSP 2007), IEEE International Conference on*, volume 4, pages 1213–1216, 2007.

[242] G. X, D. Zhan, and Z. Zhou. Supervised nonlinear dimensionality reduction for visualization and classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 35(6):1098–1107, 2005.

[243] J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the international workshop on Workshop on multimedia information retrieval*, pages 197–206, 2007.

[244] J. S. Yi, R. Melton, J. Stasko, and J. Jacko. Dust & magnet: multivariate information visualization using a magnet metaphor. *Information Visualization*, 4(4):239–256, 2005.

[245] D. Zhang, M. Islam, and G. Lu. A review on automatic image annotation techniques. *Pattern Recognition*, 45(1):346–362, 2012.

[246] S. Zhang, Q. Tian, Q. Huang, W. Gao, and S. Li. Utilizing affective analysis for efficient movie browsing. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 1853–1856, 2009.

[247] W. Zhao, R. Chellappa, P. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys*, 35(4):399–458, 2003.

LIST OF FIGURES

ACKNOWLEDGMENTS

First and foremost, I would like to thank my supervisors, Alex Telea and Fernando Paulovich. Alex, thanks for the long meetings discussing technical and sometimes non-technical subjects. I am truly amazed by the speed of your thoughts and how you can explain their complexity so easily to your students. Also, I wonder how you can preserve your enthusiasm and your good mood in your everyday life, even after a hard working day. Fernando, thanks for the countless meetings discussing ideas and life, they were essential to keep me motivated during this period. Also, you showed to me that we can enjoy different life styles, even when working a lot. To both, thanks once more for the guidance and supervision, you have helped me to reach personal and professional levels that were only possible in my most optimistic dreams.

My dear family: Mom and Dad, I know how hard you both fight to give me this opportunity, thank you. Ká and Hugo, thanks for the kind words when listening to my complaints during this time, and also for going out and have fun with me when I needed to rest my mind. Many thanks to my relatives from Patos de Minas, Cuiabá and Rondonópolis to always encourage me with my PhD work, and my apologies for the time I was absent.

Maiti-Holiudi, my Brazilian group of roommates which I am very proud to be part of: Gabriel, José Augusto, Tácito, Maycon, Flávio, Ricardo, Cássio, Léo e Fernando, thanks for all the pleasant moments when drinking, playing, traveling, mailing and talking about work, politics, soccer and relationships.

Many thanks to the Winschoterdiep friends, whom I shared many international dinners, talking, drinking and many others wonderful moments together, especially: Sven, Alex, Laura F., Nico, Camila, Victor, Antonia, Laura K., Juan, Octavio, Emilia, Stephan, Efthymis, Jelle, Thomas and Rixt. I am very glad that I got to know all of you.

Thanks to the Brazilian friends I met in Groningen for the very relaxing time together and the Turkish friends to treat me so nicely and helpful when I had just arrived there. Enrico, I am thankful and glad to have known you during this time as well.

Rafael, a special friend, but also a cinephile, a gamer, a singer, a geek,... Thanks for all the time since the master degree, in special our time in Groningen, you and Priscila helped me a lot there ("Happiness is only real if shared").

Pablo, Dr. Gabriel, Resina/Nati, Karina Sato, Jorge, Karol and Milena, all of you have been playing a important role in my life and for sure we will keep like this in the future. Jonas, Aninha, Mark and Léo, old friendships like ours are memorable, thanks!

My research group in Brazil VICG (and ICMC in general), thank you for discussing useful (or not) ideas while drinking coffee, and moments of making

fun of each others inside the office. Thanks go to the SVCG research group in Groningen, that treat me kindly and helped me to improve my research skills. Many thanks to the 5th floor JBI friends, for exploring the city, countries and restaurants, and spending great moments together. Also, I would like to acknowledge the research funding provided by the Brazilian agencies FAPESP (grant 2011/17925-1), CNPq (grant 156580/2011-0) and the research project CAPES/NUFFIC 028/11.

Carol, for believing in me more than I usually do, giving me support in hard times and sharing so many great moments together: Thanks a lot.

God: No matter who, what or where are You. No matter if You are only one or many. No matter which religion You really belong (if You, in fact, belong to some). What matters is that you are providing me wonderful moments, in wonderful places, with wonderful people. Thanks!

ABOUT THE AUTHOR

Danilo B. Coimbra was born in Ribeirão Preto, Brazil in 1985. His curiosity about technologies and portable electronic devices was the trigger to his interest in computers. After doing part of his MSc by also pursuing a half-time job, he decided to follow a full-time PhD track to full dedicate to academic research. This was realized by a so-called sandwich PhD project, which led to a double PhD degree between the Universities of São Paulo and Groningen.

Since the beginning of his studies in Computer Science, his strong interest was to help a broad variety of users, providing tools, software, and techniques that could help people beyond specialist users for specific subjects. While studying towards his BSc degree in Computer Science from the COC College, he worked in a supply chain company, developing Web solutions to improve the logistic process, such as routing algorithms. In this context, he had his first foreign work experience, when he went to Montevideo, Uruguay, to manage a software installation and test activity procedures. In his final BSc project, he developed an e-learning tool aimed to use ubiquitous computing concepts to help professors when creating virtual assignments for their students.

With the advent of online streaming multimedia content all over the internet, in combination with his MSc subject on Video Analysis, he saw an opportunity to develop a video segmentation technique that provides more semantic and meaningful video segments for the TV news genre. Motivated by the good results thereof, he decided to continue as a PhD student in the same institution (University of São Paulo). The ensuing research challenge was how to present and visualize not only multimedia content, but any kind of complex, high dimensional, data. During his PhD, he had his second work experience abroad in the Netherlands, when he learned many different aspects in terms of culture, society and research.

The eagerness to continuously learn, share (useful or less useful) knowledge, and believe that developing novel and creative solutions can really make a difference in some people's' lives, led him to his current professional and personal position, and to his very rewarding double PhD degree experience.

COLOPHON

This thesis was typeset with $\LaTeX\,2_\varepsilon$ using Robert Slimbach's *Minion Pro* font. The typesetting is based on the *classicthesis* style by André Miede.