EXPLORING AND EXPLAINING 3D DATA REPRESENTATIONS

XIAORUI ZHAI

Cover: Illustration of the similarity of 3D scatterplots created by explained dimensionality reduction (background: *Wine* dataset projected by the FA method, see Sec. 6.4.2, Fig. 6.4) and directly acquired as spatial measurements (foreground: Andromeda galaxy star cloud, see Sec. 3.4, Fig. 3.7). The thin black curves show the latter dataset's 2D skeleton which can be used for direct rotation (see Chapters 3 and 4).

Exploring and Explaining 3D Data Representations

Xiaorui Zhai
PhD Thesis

The research for this dissertation was conducted at the Scientific Visualization and Computer Graphics (SVCG) research group, part of the Bernoulli Institute (BI) and the Faculty of Science and Engineering (FSE) at the University of Groningen, the Netherlands.

university of groningen

Exploring and Explaining 3D Data Representations

PhD thesis

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

This thesis will be defended in public on

Thursday 17 November 2022 at 11.00 hours

by

Xiaorui Zhai

born on March 10, 1982
in Henan, China

Supervisor
Prof. A. C. Telea

Co-supervisor
Dr. Lingyun Yu

Assessment committee
Prof. J. B. T. M. Roerdink
Prof. J. Kosinka
Prof. C. Hurter

ABSTRACT

Visualization is a key element of analyzing, interpreting, and understanding phenomena described by multidimensional datasets. Many visualization methods exist for such datasets with strong variations in their design and use as a function of the dimensionality of the involved data. However, several challenges concerning the exploration and explanation of multidimensional data via visualization remain open.

In this thesis, we explore these challenges with a focus on the visualization of data by means of three-dimensional (3D) data representations. We divide our contributions in this respect into two parts. First, we investigate the usage of visualizations of 3D shapes and low-dimensional data represented by 3D scatterplots. We identify the choice of a suitable viewpoint as an important challenge for this type of exploration and, in particular, the examination of such 3D representations by rotating the viewed data along a flexibly-specified 3D rotation axes. We propose a novel technique for interactively specifying 3D rotation axes that capture the local structure of the data. For this, we use the so-called skeletons computed from the 2D silhouette of the visualized shape and augment these with depth information. Our technique is easy to implement and computationally and visually scalable to large and complex 3D shapes. An evaluation study shows that our technique complements well existing 3D rotation mechanisms such as the virtual trackball.

Our second contribution regards the visual exploration of high-dimensional datasets depicted as scatterplots created by dimensionality reduction (DR). We first identify explanation of the visual structures present in such scatterplots to be a key challenge to their understanding. We improve existing visual explanation techniques by methods that consider the correlation and local dimensionality of the projected points, and demonstrate how our approaches can bring added value in explaining 2D DR scatterplots. Secondly, we study how 2D DR scatterplots, well known in the visualization literature, compare with 3D DR scatterplots, which have been far less explored. We present a quantitative study to compare 2D and 3D DR scatterplots along a rich selection of datasets, projection techniques, and quality metrics. We find that 3D projections bring only limited added-value atop of the one provided by their 2D counterparts, but they can show more structure than their 2D counterparts, and thus can stimulate users to further exploration.

SAMENVATTING

Visualisatie is een sleutelelement voor het analyseren, interpreteren en begrijpen van fenomena die beschreven worden door multidimensionale dataverzamelingen. Veel visualisatiemethodes zijn bekend voor dergelijke datasets met een grote variatie in hun ontwerp and gebruik in verband met de dimensionaliteit van de te visualiseren gegevens. Niettemin blijven verschillende uitdagingen voor de exploratie en uitleg van multidimensionale gegevens via visualisatie nog steeds open.

Dit proefschrift bestudeert deze uitdagingen met een focus op de visualisatie van gegevens via driedimensionale (3D) representaties. Onze bijdrage in deze richting kan gesplitst worden in twee delen. Eerst bestuderen wij het gebruik van visualisatie van 3D vormen en laagdimensionale gegevens afgebeeld als *scatterplots*. Ons werk vindt dat het kiezen van een geschikt kijkpunt een belangrijke uitdaging is voor dit type exploratie, vervolgd door het bestuderen van dergelijke 3D representatie door middel van rotatie van de afgebeelde data rond een flexibel gespecificeerde 3D rotatieas. We ontwikkelen een nieuwe techniek voor het interactief specificeren van 3D rotatieassen die de locale datastructuur gebruikt. Dit wordt gerealiseerd door middel van zogenaamde skeletten die worden berekend uit de 2D silhouette van de gevisualiseerde vorm en verder verrijkt met diepteinformatie. Onze techniek is makkelijk te implementeren en ook computationeel en visueel schaalbaar tot grote en complexe 3D vormen. Een evaluatiestudie laat zien dat onze techniek een goede aanvulling is voor bestaande 3D rotatiemechanismen zoals de virtuele *trackball*.

Onze tweede bijdrage betreft de visuele exploratie van hoogdimensionale datasets afgebeeld als *scatterplots* gemaakt via dimensionaliteitsreductie (DR). We laten eerst zien dat het uitleg van de visuele structuren van deze *scatterplots* een hoofduitdaging is voor het begrijpen daarvan. Wij verbeteren bestaande technieken voor visueel uitleg door het gebruik van correlatie en locale dimensionaliteit van de geprojecteerde datapunten en laten zien hoe onze verbeteringen toegevoegde waarde brengen voor het uitleg van 2D DR *scatterplots*. Vervolgens bestuderen wij hoe 2D DR *scatterplots*, reeds goed bekend in de visualisatieliteratuur, zich vergelijken met 3D DR *scatterplots* die veel minder bestudeerd zijn. We presenteren een kwantitatieve studie die 2D met 3D DR *scatterplots* vergelijkt over een brede selectie van datasets, projectietechnieken en kwaliteitsmetrieken. Onze bevindingen laten zien dat 3D projecties beperkte toegevoegde waarde brengen boven wat de 2D projecties kunnen doen, maar, aan de andere kant, beelden meer structuur af dan 2D projecties en dus gebruikers kunnen stimuleren tot verdere exploratie.

PUBLICATIONS

This thesis is the result of the following publications:

- Interactive Axis-Based 3D Rotation Specification Using Image Skeletons (Zhai et al., 2020)
- Skeleton-and-Trackball Interactive Rotation Specification for 3D Scenes (Zhai et al., 2022)
- Using Multiple Attribute-Based Explanations of Multidimensional Projections to Explore High-Dimensional Data (Tian et al., 2021c)
- Quantitative and Qualitative Comparison of 2D and 3D Projection Techniques for High-Dimensional Data (Tian et al., 2021d)
- Enhanced Attribute-Based Explanations of Multidimensional Projections (van Driel et al., 2020)

CONTENTS

1 INTRODUCTION

Multidimensional data is increasingly more present in scientific research and industrial practice. Such data can be seen as a set of observations (also called samples or measurements), each consisting of several measured quantities (also called dimensions or variables or attributes). The increase of quality and availability of sensing devices, simulation systems, and storage solutions have led to the current context when applications generate so-called big data collections consisting of millions of observations each having tens up to thousands of dimensions.

Visualization is a key element of any data analysis process and several methods exist for multidimensional data visualization. At a high level, regardless of the nature and type of the studied dataset, all these methods aim at the same goal: Provide efficient and effective ways for stakeholders interested in the problem domain from which the dataset has emerged to study the respective dataset and find interesting patterns in the data. To achieve this, visualization methods select various ways to *encode* the dataset's attributes into so-called visual variables, such as position, size, color, transparency, or texture. Next, the stakeholders involved in studying the visualization decode the patterns present in the data by interpreting the visual patterns created by these visual variables. During the exploration process, besides the above-mentioned data encoding choices of the visualization method, *interaction* is the second key element of such visualization methods. Interaction allows users to select which parts of the data to examine and how to parameterize the data encoding or, more globally put, how to *look* at the data.

While present in virtually any visualization method, the two above-mentioned key ingredients of visualization methods – data encoding and interaction – differ significantly depending on the dimensionality of the dataset at hand. We make a distinction between low-dimensional and high-dimensional datasets, and their challenges for visualization, as follows.

1.1 LOW-DIMENSIONAL DATASETS

At one end of the spectrum we have low-dimensional datasets consisting of a few, roughly two to four, dimensions. Unsurprisingly, most visualization methods for such datasets select spatial coordinates as the main visual variables to map data to. Within such low-dimensional datasets, we can further find a distinction between inherently spatial datasets and abstract datasets, as follows.

Inherently spatial datasets contain attributes which represent the measurement of actual positions in the two-dimensional (2D) or three-dimensional (3D) space. Examples hereof are 2D or 3D shapes modeled by boundary representations (meshes) or volumetric representations. Such datasets emerge from many application domains such as 3D modeling or 3D scanning present in engineering or the media industry; numerical simulations of physical processes present in various fields of science; and volumetric acquisition techniques such as CT or MRI scans used in medical science. For such inherently spatial datasets, the data to visual variable mapping is straightforward and natural: The spatial dimensions present in the dataset are directly mapped to the spatial dimensions of the visualization. Additional attributes present in the dataset can be mapped to visual variables beyond spatial ones, such as color, texture, or transparency. Such inherently spatial datasets form the topic of a separate subfield called scientific visualization or *scivis* (Telea, 2014b; Yu et al., 2010; Jackson et al., 2013).

Depending on the continuity properties of the underlying data, the resulting visualization can depict compact shapes (such as in the case of the aforementioned 3D surfaces acquired by modeling or scanning) or discrete collections of points. The latter case leads to visualizations typically known as *point clouds* (Liu et al., 2021; Yu et al., 2012) or *scatterplots*. Finally, an important interaction element for such visualizations is allowing users to change the *viewpoint* to examine and explore these inherently spatial 3D datasets from various angles.

Abstract datasets contain attributes which do not represent the measurement of actual spatial positions. Rather, such attributes can come from any physical or non-physical domain. Such datasets are most often represented as tabular collections of measurements performed on a given population; table rows represent the individual samples taken (also called observations or data points); table columns represent the different independent measurements (or variables) collected for each observation. Examples of such datasets are medical data collected on sets of patients such as gathered by electronic patient dossiers; economical data gathering a number of indicators measured over a given set of actors; or data gathered from the simulation or observation of astronomical phenomena. For such datasets, the mapping from data to visual variables is less straightforward than for their inherently spatial counterparts: One has to decide which of the data dimensions get mapped to the visual spatial variables (2D or 3D coordinates in the visualization) and which data dimensions get mapped to other visual variables such as color.

Visualization techniques aimed at abstract datasets are studied in the context of the separate subfield of information visualization or *infovis* (Munzner, 2014; Yi et al., 2005). Typical visualizations for such low-dimensional, abstract, datasets involve 2D or 3D scatterplots similar in construction to those used to visualize inherently spatial datasets. Also,

similar to the inherently spatial case, interaction is a key element that allows users to examine such scatterplots from different angles, thereby helping tasks such as finding data clusters, outlier samples, or correlations between the mapped dimensions.

1.2 HIGH-DIMENSIONAL DATASETS

At the other end of the spectrum concerning data dimensionality, we have so-called high-dimensional datasets. While no universally accepted formal distinction between low and high dimensional datasets exists, a practical and useful characterization considers datasets to be high-dimensional from the moment when there are not sufficient independent visual variables, such as position, size, color, and the other ones mentioned earlier in this chapter, to encode all the data dimensions. Following this definition, datasets having more than roughly 5 to 10 dimensions are regarded as high-dimensional. To visualize such datasets, specific methods have been developed in the subfield of infovis. Well-known examples of such high-dimensional data visualization methods include scatterplot matrices (Yates et al., 2014), parallel coordinate plots (Inselberg and Dimsdale, 1990), and table lenses (Telea, 2006).

Dimensionality reduction (DR) methods, also known as embeddings or projections, are a special class of methods for visualizing multidimensional data (Nonato and Aupetit (2018); Espadoto et al. (2019)). Compared to other methods for the same task, such as scatterplot matrices or parallel coordinate plots, DR methods have a significant *data scalability* advantage, as they are able to depict datasets having hundreds of thousands of samples, each with hundreds of dimensions or more. DR methods use the scatterplot visualization metaphor: Every data point is mapped to a 2D or 3D visual point, regardless of the number of dimensions. Next, similar to the usage of scatterplots for low-dimensional data, users examine the resulting scatterplots to find clusters of samples, outliers, or other interesting patterns to the problem at hand.

1.3 THE EXPLORATION CHALLENGE

In the previous sections, we have introduced several types of visualization techniques for low-dimensional, respectively high-dimensional data. While such methods are very different, we believe that their main challenge is to enable an easy *understanding* of the depicted data. In turn, this requires that such methods provide suitable techniques for the user to *explore* the depicted data to reach the desired understanding.

The exploration mechanisms proposed by low-dimensional and high-dimensional visualizations have been extensively studied in separation within the scivis and infovis domains. However, relatively little work

aims to compare and contrast such mechanisms. We aim to take a step in this direction and compare – and, to a certain extent, contribute to solving – the challenges faced by low-dimensional and high-dimensional visualizations. Since this is a huge endeavor, we will need to scope the research question next, as follows.

We proceed for this by firstly choosing visualization *metaphors* to study. For low-dimensional datasets, we consider both 3D meshes and 3D point clouds visualized by scatterplots, such as present in scivis. For high-dimensional datasets, we consider DR techniques since, as explained earlier, these are some of the most scalable approaches to visualizing high-dimensional data. Moreover, DR techniques also use the scatterplot metaphor that is used to depict 3D point clouds. This limits the variety of the studied techniques, on the one hand, but makes comparing and contrasting between techniques easier.

With these preliminaries, we can now state our main research question:

How can we improve the exploration and understanding of 3D point-based visualizations encoding various types of low- and high-dimensional data?

1.4 STRUCTURE OF THIS THESIS

The work we conducted next to answer this question can be divided into two main parts, as follows.

In part 1 of the thesis (Chapters 3 and 4), we examine the exploration and understanding of low-dimensional datasets – specifically, 3D meshes and 3D point clouds. Since this data is quite low-dimensional and, in the particular case of 3D meshes, it comes from the sampling of actual shapes, we conjecture that the understanding problem is largely reduced to an *exploration* problem. Moreover, the exploration problem, for such visualizations and datasets, mainly regards the choice of a suitable *viewpoint* to examine the data shown by the 3D visualization. As such, for these datasets, we refine the global research question stated above to the following:

RQ1: How can we efficiently and effectively specify good viewpoints for the visual exploration of 3D meshes and point clouds?

We next address this research sub-question as follows. In Chapter 3, we present a mechanism for specifying 3D rotations around a wide variety of axes used in the visualization design space. In contrast to other existing techniques that aim to help users to specify such rotations by varying several axis parameters, we define the rotation axes based on the visible silhouette of the visualized shapes. This effectively allows users to grab the visualized shape at any desired point and rotate it, with

a desired angle and/or rotation speed, around the local axis of symmetry implied by the shape around the selected point. We efficiently and effectively compute such local rotation axes by leveraging the summarization power of 2D binary-image medial descriptors or skeletons. We extend these 2D skeletons with depth information to provide approximations of the abovementioned 3D local rotation axes. Our proposed technique allows one to specify such complex 3D rotations, we argue, far more easily than existing techniques, by simple point, click, and drag gestures.

As with any method aimed to help users to perform a task more effectively and/or more efficiently, our rotation specification method needs to be evaluated in a practical setting and against other methods for the same task. To address this, Chapter 4 presents a user study that we organized to compare our proposed skeleton-based 3D rotation specification mechanism with its arguably best-known, and most-used, counterpart – the virtual trackball rotation specification mechanism. Our study shows that the two techniques – skeleton-based rotation and trackball rotation – are complementary and have their own advantages and limitations depending on the type of rotation one wants to execute and the type of shape being examined. This concludes the first part of the thesis.

In part 2 of the thesis (Chapters 5 and 6), we examine the exploration and understanding of 3D scatterplot-based visualizations of high dimensional data using DR methods. Technically, DR methods create point clouds that are identical to those created from 2D or 3D scatterplots, *i.e.*, contain a set of 2D or 3D point locations. However, interpreting point clouds created by projections is far more complicated, since the axes, or spatial dimensions, of the visualization space do not have a direct meaning in terms of data dimensions. As such, the research question that pertains to such visualizations is a joint explanation and exploration one. In other words, before one can actually explore a DR projection, one needs to understand what the overall visual patterns that such a projection shows actually mean in terms of data dimensions. This aspect is not only relevant to 3D projections (represented by a 3D point cloud) but also to the more common 2D projections (represented by a 2D point cloud). As such, we refine our general research question for such visualizations to the following:

RQ2: How can we explain and explore 2D and 3D scatterplots created by dimensionality reduction techniques?

We next address this research sub-question as follows. In Chapter 5, we start with the arguably simpler case of 2D projections, which do not have the additional challenge of choosing a suitable viewpoint for examination. We address the *explanation* of these projections by developing several so-called local explanation techniques which label neighbor points in the projection by their shared data-related characteristics. The

resulting visualizations effectively split the projection point cloud into a number of differently colored and shaded zones, where each zone can be effectively explained in terms of the original data dimensions. We show how our additional explanation mechanisms complement existing explanation mechanisms and lead to a better understanding of data represented by such 2D projections.

In Chapter 6, we move to consider the more complex 3D projections and address the *exploration* part of our research sub-question. We answer this question by comparing the data-encoding abilities of 2D projections to those of 3D projections for a number of projection techniques and datasets. Our study shows that, as gauged by existing quality metrics in the projection literature, 3D projections have a measurable advantage in terms of preserving the structure of the underlying data. However, they also introduce the added complexity of choosing suitable viewpoints to examine them from. To study this aspect, we qualitatively compare the same number of 3D projection techniques run on the selected datasets with the resulting projections being annotated by the visual explanations presented in Chapter 5. This qualitative study shows that there are marked differences in what one can see, in terms of visual patterns explained by our techniques, between different projection techniques. Additionally, our study shows that users also attach a given preference to certain projection techniques depending on the ease of deducing insightful patterns from their visual explanations.

Finally, Chapter 7 summarizes our work and obtained findings regarding the original research question and also outlines potential directions for future work.

2 RELATED WORK

The main topic of this thesis concerns *exploring* complex multidimensional data by means of visualization. While these data can be of different nature, all ultimately lead to a *visual representation* that has to be used to provide explanations. As such, the nature of this visualization determines, to a large extent, how the subsequent explanations will work, and which will be the main challenges that its design will have to overcome.

As outlined in Chapter 1, our main research question concerns the design of techniques to improve the exploration and understanding of 3D point-based visualizations of both low-dimensional and high-dimensional data. We have given a concise, and necessarily limited, outline of what the elements of this research question imply in the previous chapter. However, finer points still remain to be detailed. In this chapter, we aim to provide all the necessary background to understand both the exact implications of our stated research question and the extent to which various research works in visualization have addressed these implications.

2.1 VISUALIZATION MODEL

Before we can discuss the challenges of a data visualization application, we have to define more formally how such an application operates and, in the process, also introduce other relevant notations that help us position our research scope. Note that, in this process, we will have to make some necessary simplifications to limit the extent of the discussion to aspects which are directly relevant to our research.

We begin by defining the notion of a *dataset*. Let $\mathbf{x}_i \in \mathbb{R}^n$ be a so-called sample (also called data point or observation) of some phenomenon. We call this a multidimensional sample with dimensionality $n \in \mathbb{N}$. As such, the sample can be expressed as a tuple $\mathbf{x}_i = (x_i^1, \ldots, x_i^n)$, with $x_i^j \in \mathbb{R}$, $1 \leq j \leq n$. The values x_i^j are called the dimensions of the sample (also known as variables or attributes). With this notation, let $D = \{\mathbf{x}_i\} \subset \mathbb{R}^n$ be a collection of such samples. We call D a dataset of n-dimensional samples. The assumption is that all samples in a dataset D are obtained, by actual measurement or simulation, from the same phenomenon. Hence, D describes a sampling of such an underlying phenomenon. Note that samples, and hence datasets, can in general take values which are not real numbers, *e.g.*, ordinal, categorical, or relational. Although our work can – with suitable modifications – also target such

more general cases, we will, for simplicity, not concern ourselves with such datasets.

A *visualization* technique for a class of datasets can be seen as a function, or mapping, from all possible sets $\mathcal{D}$ whose elements are datasets of the type D, to a visual representation. For a given dataset D, we can describe its visualization $V(D) = I$ as yielding a color image I that encodes the information in D. Good visualization techniques need to obey a number of properties (Telea, 2014b). Briefly put, the key to these properties is that the user of the visualization has to be able to mentally *invert* the function V by inferring properties of D from the image $V(D)$. In general, an exact mathematical inverse of the function V does not and cannot exist, given both the limits of the users' perception and various design decisions of V itself which lose information from D when creating $V(D)$. As such, in practice, a good visualization has to allow users to reasonably well quantify a subset of properties of D from its visualization $V(D)$. Figure 2.1 illustrates both the direct mapping (from D to $V(D)$) and the inverse mapping, performed by the visualization user.

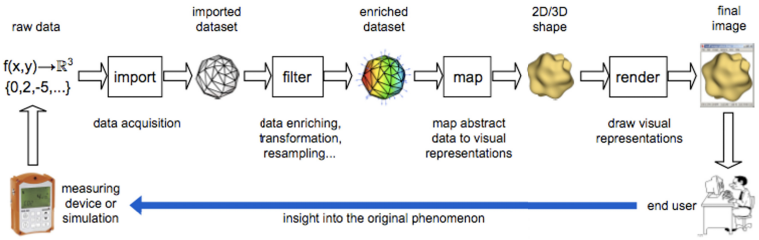


Figure 2.1: Visualization pipeline and the concept of inverse mapping. Image taken from Telea (2014b).

Discussing visualization methods V at the above generality level is hard since there is no explicit relationship between the data samples $\mathbf{x}_i$ of D and the produced image $V(D)$. However, there exists a subclass of visualizations where V can be characterized in more detail. These visualizations independently map the individual samples $\mathbf{x}_i$ via the function V to individual, recognizable, visual elements $V(\mathbf{x}_i) \in \mathbb{R}^d$. Here, d represents the number of so-called visual variables that V uses to encode (some of) the data dimensions of $\mathbf{x}_i$. Examples of these visual variables are position (of the visual element $V(\mathbf{x}_i)$ in the visualization), color, transparency, size, orientation, and texture. As such, the visualization of a dataset is the (visual) union of the visualizations of its samples, or $V(D) = \cup_{\mathbf{x}_i \in D} V(\mathbf{x}_i)$. Many examples of such visualizations exist. In our work, we will study two main examples of such sample-based visualizations, namely scatterplots, which map a data point $\mathbf{x}_i \in D$ to a 2D or 3D visual point $V(\mathbf{x}_i) \in \mathbb{R}^d$, $d \in \{2, 3\}$, and surfaces, which map a local surface element (splat or polygon) $\mathbf{x}_i \in D$ to the 3D shaded representation thereof $V(\mathbf{x}_i) \in \mathbb{R}^3$. In general, the dimensionality d of the visual space

is limited to about 8 visual variables. Defining more visual variables is technically possible but will cause serious confusions, or interference, when performing the inverse mapping outlined above. In contrast, the data dimensionality n is not limited.

2.2 TYPES OF DATA IN VISUALIZATION APPLICATIONS

The previous section has shown that the visualization pipeline, or function V, can be described by a mapping between the data space (a subset of $\mathbb{R}^n$) and the visual space (a subset of $\mathbb{R}^d$, where d is upper bounded by perception). As n can be equal, higher, or much higher than d, designing suitable visualization techniques V incurs different challenges. We will thus classify and discuss such challenges as a function of the dimensionality n of the input data (since the dimensionality d of the visual space is limited, as explained above). In this sense, we see two main cases, as follows.

Low-dimensional data. This situation describes datasets whose dimensionality n is below or equal to the visual dimensionality d. That is, every data dimension can be mapped to a different visual dimension. The key advantage of this situation is that *decoding* the visual dimensions, *i.e.*, going from the visualization back to the data properties, can be done relatively easily – each data dimension is, after all, reflected in a different visual stimulus.

Within this class of datasets, visualization techniques typically make a subsequent distinction between spatial data and non-spatial data. Given that this distinction is important in visualization literature and practice, we will next discuss it and, more importantly, we will discuss why we believe this distinction is not crucial to our own research question.

Spatial data refers to data which is, by its nature, measured (sampled) at locations in 2D or 3D, such as scalar or vector fields, images, CT or MRI volumes, and scanned or computer-generated geometric shapes. Given this property, the data samples $\mathbf{x}_i$ can be *directly* mapped, one-to-one, to visual locations in the visualization. As such, one literally 'sees' the structure of the data in the visualization. Additional data attributes (beyond the sample locations, that is, *e.g.*, pressure and velocity in a fluid flow simulation or tissue density in a CT scan) can be mapped to visual variables beyond position such as color, size, or transparency. Visualization techniques for spatial data fall into the realm of *scientific visualization* or scivis.

Still, this does not mean that explaining or exploring such data is trivial. In the particular case of 3D data depicted by a 3D visual representation, a simple to explain but not simple to solve problem is where from to *look* at the data. Choosing good viewpoints is important especially in the case of complex datasets, *e.g.*, complex shapes with intri-

cate structure. No single viewpoint can, in general, suffice since views of such datasets inherently suffer from self occlusion due to their complex structure. As such, choosing suitable viewpoints is important. We discuss this further in Sec. 2.4.

Non-spatial (abstract) data refers to data which is, by its nature, not defined as the sampling of a phenomenon naturally living in 2D or 3D. However, by the very definition of visualization, the visual representations of such data have to live in 2D or 3D to be visible and visually explorable. For the low-dimensional case ($n \leq d$), such datasets are typically tables of d dimensions, where each column is the sampling of a dimension and each row a separate sample $\mathbf{x}_i$, respectively. The nature of the dimensions (columns) need not to be directly spatial, such as *e.g.* if one considers a table where rows (samples) are persons from an opinion poll and columns capture the subjects' age, salary, and voting intentions. Such low-dimensional datasets can be visualized by directly mapping their n data attributes to 2D or 3D coordinates, plus several other visual variables such as size, color, or annotations. Probably the best-known example of this type of visualization is the bubble chart which generalizes classical scatterplots (which encode two or three data dimensions in spatial dimensions) by using the size, color, and transparency of the visual shapes used to depict the data points. Visualization techniques for non-spatial data fall into the realm of *information visualization* or infovis.

Such low-dimensional abstract data visualizations share many commonalities with the low-dimensional visualizations for spatial data. Most importantly, every data dimension is mapped to a separate visual dimension, hence the decoding (inverse mapping) of the visualization is relatively straightforward. In other words, one can directly assign meaning to the visual *axes* of the visualization – each such axis encodes one of the data dimensions. For 3D visualizations, however, the same main challenge exists here as for spatial data, namely picking a good viewpoint (or viewpoints) to understand the data structure. Note that this viewpoint-picking challenge is relatively higher for non-spatial data visualizations. Indeed, while visualizing spatial data shows some arguably familiar shapes to the user (such as a 3D scan or brain CT surface) that can be understood by the professional user from few viewpoints, visualizing non-spatial data shows a more abstract *scatterplot* whose patterns are, in general, harder to understand.

High-dimensional data. Apart from the above, one has data which consists of a higher number of dimensions n than d. When the number of dimensions is (significantly) larger, or $n \gg d$, the individual data dimensions cannot be any more mapped to individual visual variables – there simply are not enough distinct visual variables for that. A well known solution for this is to use techniques that reduce the data dimen-

sions to visual dimensions in such a way that important properties of the data are preserved by the visualization.

There are many techniques in the information visualization domain that address such high-dimensional data. At a general level, all such techniques have to perform a *data reduction* to fit all the n data dimensions into the available d visual variables. We outline below a few well-known such techniques with the focus on the one that we choose to explore in our work next.

Table lens techniques (Telea, 2006) essentially overload the visual spatial dimensions in d to show multiple data dimensions in n. Simply put, table lenses plot a dataset D as a matrix where each row is a sample $\mathbf{x}_i$ and each column is a dimension in n, respectively. The values x_i^j of the samples' dimensions are encoded by table cell visual attributes such as color or transparency. An important added value of table lens techniques is their ability to *aggregate* multiple samples $\mathbf{x}_i$ and display them minimally, *e.g.*, as a single row of colored pixels. Interaction mechanisms next allow the user to specify which samples of a dataset D can be aggregated. This effectively handles the problem of datasets D having many samples. Still, each of the n dimensions needs its own visual space (table column) to be displayed.

Scatterplot matrices (Becker et al., 1996) also overload the spatial dimensions in d, but in a different way. They use a so-called small-multiple design to show N 2D scatterplots of pairs of dimensions (x^i, x^j), $1 \leq i \leq n$, $1 \leq j \leq n$, of the dataset D. Scatterplot matrices scale well to the number of samples in D but quite poorly (quadratically) to the number of data dimensions n. Moreover, relatively involved interaction is needed to show which points of interest in a scatterplot correspond to points in all of the other scatterplots in the matrix.

Parallel coordinate plots (Inselberg and Dimsdale, 1990) also overload the spatial dimensions in d. Every dimension in n gets a separate axis, or coordinate, to plot its samples. A sample $\mathbf{x}_i$ is then recognized visually as a *polyline* that connects its different values x_i^j among all the n axes. Parallel coordinate plots scale similarly to table lenses in the number of data dimensions n but better in the number of data samples. However, the latter scaling creates overplotting, that is, multiple polylines corresponding to multiple samples overlap in the same visual space.

Dimensionality reduction (DR) methods (Espadoto et al., 2019) take a quite different approach to map the n data dimensions to the d visual variables. Simply put, DR methods create a low-dimensional (typically, 2D or 3D) scatterplot in which every point $V(\mathbf{x}_i)$ is the result of mapping a data point $\mathbf{x}_i$. Key to their working is that they aim to set the visual distance between scatterplot points $\|V(\mathbf{x}_i) - V(\mathbf{x}_j)\|$ to reflect as closely as possible the distance between the corresponding data points $\|\mathbf{x}_i - \mathbf{x}_j\|$. Various heuristics are used for this so-called 'preservation of the data structure', as reflected by many DR methods such as PCA (Jolliffe, 2002), t-SNE (van der Maaten and Hinton, 2008), or UMAP (McInnes et al.,

2018), to mention only the best known ones. Dimensionality reduction is by definition an ill-posed problem as it is, in general, not possible to preserve all above-mentioned pairwise distances between data points in the distances between their visual counterparts. As such, many DR methods have been created which aim to preserve various aspects of the so-called higher-level 'data structure' in the visualization. Elaborate studies on the relative success of such methods to preserve such data structure, according to various quality metrics, are present in the DR literature (Espadoto et al., 2019; Nonato and Aupetit, 2018; Kehrer and Hauser, 2013; Cunningham and Ghahramani, 2015; Sorzano et al., 2014). On the other hand, DR methods are *far* more scalable than other methods for visualizing high-dimensional data such as scatterplot matrices, table lenses, or parallel coordinate plots. Indeed, in any DR method, *any* number of data dimensions n is 'collapsed' to the available two or three visual spatial dimensions given by a 2D, respectively 3D, scatterplot. Additionally, every sample $\mathbf{x}_i$ in a DR plot is mapped to a single point (in the limit, pixel) in the visualization. This strongly contrasts the significantly larger visual space needed by other methods to map such a sample – a horizontal pixel row for table lenses; $n^2/2$ points in the respective pairwise 2D scatterplots for scatterplot matrices; and an entire polyline for parallel coordinate plots.

Apart from using the position visual variables to encode data dimensions, DR methods can naturally use other visual variables (color, point size, texture) to encode additional data dimensions. In this sense, DR methods are quite similar to the 2D or 3D scatterplots discussed above for low-dimensional non-spatial data. However, there is a major difference between scatterplots created from low-dimensional data and those created by DR methods: The former, as explained earlier, explicitly encode one of the n data dimensions in a visual spatial dimension in the d-dimensional visual space. DR methods, as they have to handle many more data dimensions, encode *sets* of these dimensions in the visual spatial dimensions in the d dimensional visual space. As such, *interpreting* a DR scatterplot is far harder than interpreting a 'classical' scatterplot for low-dimensional data: For the DR case, the visual spatial dimensions cannot be directly mapped to data dimensions. To give arguably the simplest example, the x and y visual spatial dimensions of a 2D PCA scatterplot encode, each, linear combinations of *all* the n data dimensions that the projection has worked on.

As such, the task of *explaining* DR plots gains strong significance since, without suitable explanations, the user cannot readily interpret the visual patterns such a plot presents. Many methods exist for such DR explanatory tasks. Recently, local techniques have been proposed to annotate groups of close points in the projection by the identity of the data dimension in the n ones that makes such points similar. We discuss this class of techniques, which we choose to further explore

and improve, as well as other explanatory techniques for DR plots in Sec. 2.6.

2.3 SCOPE OF INVESTIGATIONS

Summarizing the above review of item-based visualization methods for both low-dimensional and high-dimensional data, we draw the following general conclusions:

- A main challenge for *low-dimensional* data visualizations, whether of spatial or non-spatial data, is to choose *good viewpoints* to examine the respective visual mappings. This challenge is most important for 3D visualizations which allow multiple viewpoints for their exploration.
- Visualizations of high-dimensional datasets is arguably best served, in terms of visual scalability, by dimensionality reduction (DR) methods.
- A main challenge for *high-dimensional* data visualizations, which are typically by their nature handling non-spatial data, is to *explain* what the visual patterns shown in a projection mean.
- The challenge of choosing a *good viewpoint* to explore a visualization is quite similar for visualizations of low-dimensional data and high-dimensional data (that is, choosing a good viewpoint is hard for both situations, as long as both visualizations use a 3D scatterplot metaphor).

We next discuss existing methods to address these specific challenges – that is, choosing a good viewpoint for exploring 3D scatterplots of data and explaining 3D scatterplots of data created by DR methods.

2.4 EXPLORING BY CHANGING THE VIEWPOINT

In order to choose good viewpoints for visualizations of spatial data or low-dimensional abstract data, one has to essentially control all parameters that specify such a viewpoint. Formally put, a 3D viewpoint can be defined as a tuple $\mathcal{V} = (\mathbf{x}, \mathbf{v}, \mathbf{u}, \mathcal{P})$ consisting of a point $\mathbf{x} \in \mathbb{R}^3$ where the viewer is located; a view direction $\mathbf{v} \in \mathbb{R}^3$; a so-called up vector $\mathbf{u} \in \mathbb{R}^3$ contained in the view (or projection) plane and indicating the vertical direction in that plane; and a projection function P, typically set to perspective projection or orthographic projection (Marschner and Shirley, 2021). The specification of P is typically done by using control keys, given that, in most cases, this involves a choice from a predefined set of projections. As such, the challenge of interaction concentrates mainly in specifying $\mathbf{x}$, $\mathbf{v}$, and $\mathbf{u}$.

In general, fully specifying all components of $\mathbf{x}$, $\mathbf{v}$, and $\mathbf{u}$ is complex to be done interactively and not needed for *exploring* a 3D shape or scene. As such, typical interaction techniques restrict themselves to specifying a constrained subset of the space in which these three vectors can vary. Moreover, one can note that this specification amounts to giving a transformation from the canonical 3D coordinate system (in which the 3D shape to be explored is defined) and the coordinate system defined by the vectors ($\mathbf{v}$, $\mathbf{u}$, $\mathbf{v}\times\mathbf{u}$) and having as origin the point $\mathbf{x}$. In turn, such a transformation can be defined as the composition of a translation T and a rotation R.

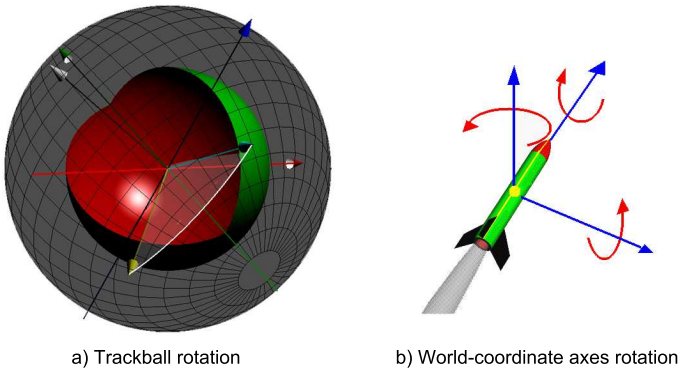


Figure 2.2: Virtual trackball rotation (a) and rotation around the world coordinate axes (b). Left image taken from Henriksen et al. (2004b). Right image taken from NASA (2022).

Specifying the *rotation* component R is significantly harder than specifying the translation T, given that R has in general 7 degrees of freedom while T has only 3 degrees of freedom[1]. Two main 3D rotation types exist – a rotation around a *center* and rotation around an *axis* – at least, if we are to stay concerned with how *users* employ existing rotation-specification mechanisms, leaving away the technicalities of how rotation is actually *implemented*.

Both rotation types are important for exploration tasks, *e.g.*, when examining a 3D shape from multiple viewpoints. Rotations around a center can be easily specified via classical (mouse-and-keyboard) or touch interfaces by well-known metaphors such as the virtual trackball (Fig. 2.2a) as described by Chen et al. (1998) and Jackson et al. (2013).

1 From a formal viewpoint, one can specify a line using fewer than 7 degrees of freedom. Minimally put, a line *direction* in 3D is a 3D vector which requires only 3 degrees of freedom, with the rotation along that vector being a fourth degree of freedom. Yet, in practice, specifying such minimal information is cumbersome and non-intuitive for the typical user. Existing tools allow a 3D line-and-rotation-along specification mainly by specifying two 3D points and a rotation, which amounts to 7 degrees of freedom.

Virtual trackball mechanisms effectively simulate the so-called 'shape in hand' metaphor where the user looks at a shape held in hand from multiple viewpoints. In line with what we stated above, trackball rotation mechanisms can be thought of using a rotation axis; however, for all practical purposes, users are not exposed to this, and as such, we can claim that, from an user perspective, the actual location (and depiction and specification) of an exact rotation axis is irrelevant to the trackball mechanism.

Rotations around an *axis* are also simple to specify if the rotation axis coincides with one of the world-coordinate axes (Fig. 2.2b) as described by Duffin and Barrett (1994). In contrast, rotations about arbitrary 3D axes are significantly more complex to specify since these involve the full 7 degrees of freedom (six degrees to specify the 3D rotation axis and an additional degree specifying the rotation angle around this axis). To picture such a situation, one can imagine that the rocket shape in Fig. 2.2b) were defined in a different world coordinate system than the one shown in that figure and one would still want to rotate the shape around its main symmetry axis.

We further discuss interactive mechanisms for specifying 3D rotations for viewpoint manipulation purposes, including their advantages and limitations, in the context of our own work in this area in Sec. 3.2.

2.5 EXPLORING BY REDUCING DIMENSIONALITY

As discussed earlier in Sec. 2.2, when the number of data dimensions n exceed the number of available visual variables d, one needs to somehow *reduce* the information captured by the n dimensions to the displayable 'bandwidth' of the d visual variables. As also outlined there, dimensionality reduction (DR) methods, also known as projections, are a particularly successful class of techniques for this task, as they are able to handle datasets having both very large number of samples and dimensions. Figure 2.3 illustrates a typical use of 2D projections to explore a high-dimensional dataset, FashionMNIST (Xiao et al., 2017), consisting of 10K samples, where each sample is a 28×28 grayscale image of a fashion item (so the dataset has a dimensionality $d = 784$). Each sample (image) is depicted as a point and colored by the class of the respective sample (out of a total of 10 classes). The left image (Fig. 2.3a) shows the dataset's projection using the by now famous t-SNE (van der Maaten and Hinton, 2008) projection technique. The right image (Fig. 2.3b) shows the same dataset projected with a sparse variant of one of the earliest DR techniques, Principal Component Analysis (PCA) (Zou et al., 2006). As visible from the figure, the t-SNE projection shows clusters of similar-colored points, *i.e.*, conveys the insight that similar-looking images in the dataset are of the same type (class). PCA shows a similar insight but the visual separation of same-color clusters is far less pronounced.

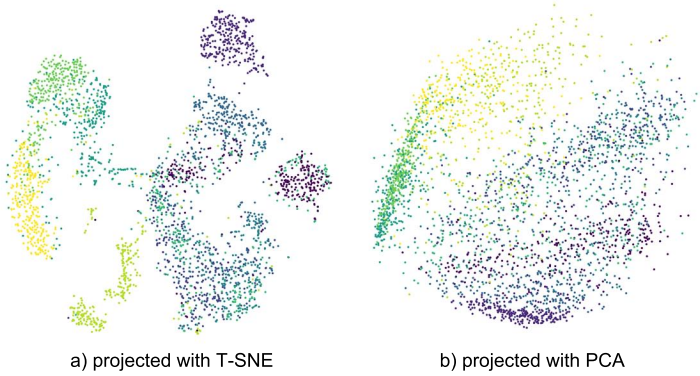


Figure 2.3: FashionMNIST (Xiao et al., 2017) dataset projected by (a) t-SNE and (b) PCA dimensionality-reduction techniques.

Since Principal Component Analysis (PCA) was first proposed, tens of different projection techniques have been developed, offering many options to data scientists, but also the added challenge in choosing a suitable technique for their goals (Nonato and Aupetit, 2018). Choosing a suitable projection technique for a given context (application, task, or dataset) is critical since, even for the same dataset, different techniques yield different visualizations, thus leading to potentially different insights and courses of action in the underlying problem solving. This issue, well recognized in the infovis and Machine Learning (ML) communities, has been mainly addressed by surveys that compare projection techniques from various perspectives. Sorzano et al. (2014) discussed 30 such techniques with a focus on optimization heuristics and cost functions used by the underlying projection algorithms. Heulot et al. (2017) proposed a taxonomy of the types of errors generated by projection techniques. Kehrer and Hauser (2013) proposed a different taxonomy to capture the tasks that DR addressed. Nonato and Aupetit (2018) surveyed the use of 28 projections in visual analytics (VA) tasks, and categorized these based on the type of errors that they produce and their effect on the performed tasks, thereby extending insights from Heulot et al. (2017) and Kehrer and Hauser (2013). More recently, Espadoto et al. (2019) presented the most comprehensive, to our knowledge, quantitative evaluation of projections, which included 44 techniques evaluated against 7 quality metrics over 18 datasets.

However, most of the above surveys consider only 2D projections, *i.e.*, projections that create a scatterplot $V(D) \subset \mathbb{R}^2$. 3D projections, which create a scatterplot $V(D) \subset \mathbb{R}^3$, are discussed only rarely. There are several reasons for this. First, surveys that consider the taxonomy of projection *algorithms* or *tasks* supported by projections arguably do

not have to make a distinction between 2D projections and 3D projections, since (a) the vast majority of projection algorithms operate similarly (if not identically) whether their target space is $\mathbb{R}^2$ or $\mathbb{R}^3$; and (b) typical tasks that projections address, such as the identification of compact clusters of sample points, correlation of these clusters with data labels, or finding outlier points, should be supported by both 2D and 3D projections. Secondly, 2D projections are far more present in actual application scenarios, as they are (a) easier to use (they do not require complex mechanisms for choosing viewpoints in 3D), (b) easier to communicate about (one can easily include snapshots of 2D projections in a website or paper since there are no viewpoints to choose from), and (c) easier to interpret (there is no depth component of the resulting image to reason about). Several authors also argue that 2D projections pose fewer interpretation and exploration challenges than their 3D counterparts (Newby, 2002; Westerman et al., 2005; Sedlmair et al., 2013).

However, 3D projections also arguably offer advantages in comparison to their 2D counterparts. First and foremost, reducing the n data dimensions to $d = 3$ visual dimensions is a smaller 'drop' in dimensionality than when going from the same n to $d = 2$. As such, 3D projections should be able to better preserve data patterns, a fact verified empirically by Coimbra et al. (2016) and separately by Poco et al. (2011) on a few datasets and projection techniques. Figure 2.4, taken from Coimbra et al. (2016), illustrates this for a 12-dimensional dataset describing software systems. In this dataset, every sample is a software project (around 6000 in total); its 12 dimensions are average software quality metrics computed for the respective project. Image (a) shows the dataset projected to 2D using the LAMP (Joia et al., 2011) projection technique. One can easily distinguish two well separated clusters of points A' and B' in the projection which, upon closer investigation, show to contain large libraries and applications, respectively small software systems. Image (b) shows the same dataset projected to 3D using also LAMP. In contrast to the 2D projection, we now see *three* clusters which, upon investigation, show to contain large libraries (A), large applications (B), and small systems (C). Hence, the use of a 3D projection enabled us to split the dataset into finer-grained sample groups.

Other arguments in favor of 3D projections include the observation that such methods are better at encoding data structure for datasets with intrinsic dimensionality exceeding three (Jolliffe, 2002). A separate argument that we outline – and which, to our knowledge, has not been systematically explored – is that 3D projections are able to create a richer set of visual patterns than their 2D counterparts and, as such, have the potential to better capture the underlying data structure. More related work concerning the evaluation of both 2D and 3D projections and also their relative advantages and limitations is discussed in the context of Chapter 6.

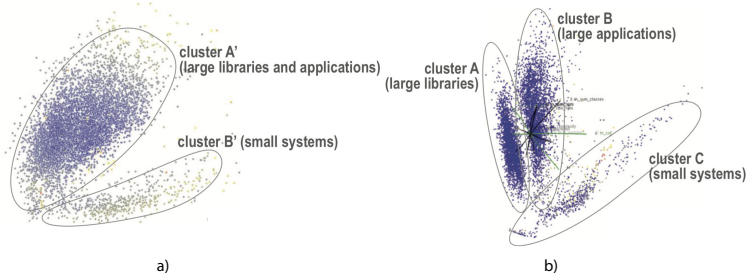


Figure 2.4: Software quality dataset projected to (a) 2D and (b) 3D using the LAMP technique. Figure taken from Coimbra et al. (2016).

2.6 EXPLAINING THE REDUCED DIMENSIONALITY

As outlined in Sec. 2.2, 'raw' projections, whether 2D or 3D ones, are of little use as one may see certain visual patterns, *e.g.*, clusters of points or outlier points, but does not know what these visual patterns *mean* in terms of underlying data patterns. To address this issue, several so-called *explanatory* techniques have been proposed for DR projection methods. At a high level, these techniques enrich the raw projection scatterplot with additional information aiming to connect the scatterplot (or parts thereof) with the original data dimensions (or parts thereof). We discuss below several such explanatory techniques. All techniques are illustrated in Fig 2.5 (taken from Coimbra et al. (2016)) using the same software dataset described earlier in Sec. 2.5.

Color coding: Arguably the simplest and most frequently used explanatory technique for DR scatterplots colors the points $V(\mathbf{x}_i)$ by the values x_i^j of a dimension $1 \leq j \leq n$ of the projected dataset. The color gradients one can next observe in the projection, if correlated with the visual point groups exhibited by the projection, allow one to explain such groups in terms of their values for dimension j. For example, in Fig. 2.5, the 3D projection is colored by the values of the *ln-cof* attribute (which describes how strongly files in a software project are coupled with each other). The projection exhibits three well-separated point clusters denoted A, B, and C in the figure. One can see that A is blue, *i.e.*, contains samples having a low coupling value. In contrast, clusters B and C are gray and yellow, denoting samples having a medium, respectively medium-to-high, coupling value. Hence, the *ln-cof* attribute is useful in distinguishing cluster A from the other two.

However effective and easy to use, and applicable to both 2D and 3D projections, color coding asks the user to manually select in turn all the dimensions j of a dataset to color code the projection and detect interesting patterns. As such, it does not scale well for datasets having

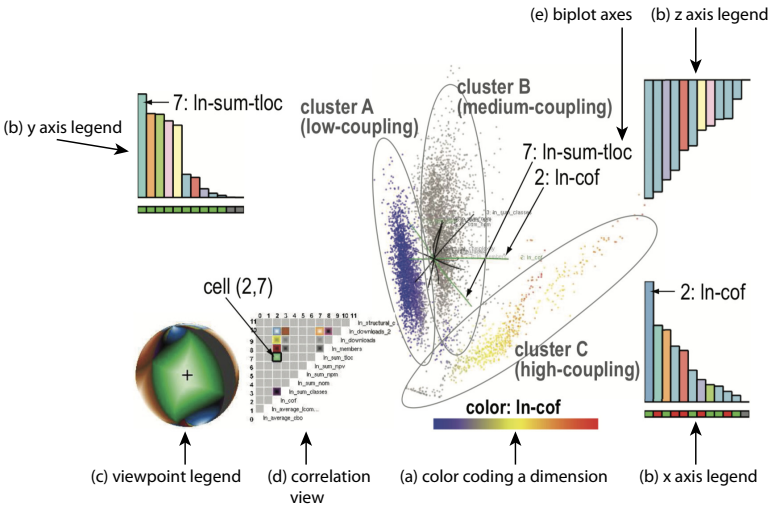


Figure 2.5: Explanatory techniques for the 3D projection of a software dataset. (a) Color coding a dimension. (b) Axis legends. (c) Viewpoint legend. (d) Correlation view. (e) Biplot axes. Figure taken from Coimbra et al. (2016).

more than 5 to 8 dimensions.

Biplot axes: Another simple and generic explanatory technique is given by biplot axes (Gower and Hand, 1995; Greenacre, 2010). In their original implementation, these are lines drawn atop a 2D projection scatterplot which indicate the directions of maximal variation of every of the n dimensions in the projection plane. They can be seen as generalizing the usual Cartesian plot x and y axes in helping the reader understand in which directions do each of the data dimensions vary the most. Coimbra et al. (2016) generalized the straight-line biplot axes (which were originally computed only for linear projections) to curved biplot axes and thereby made them applicable to any (linear or nonlinear) projection technique. They also applied them to annotate 3D projections. Figure 2.5 shows, in the center of the scatterplot, these biplot axes, two of which are annotated to indicate that they correspond to the dimensions *ln-sum-tloc* and *ln-cof* of the dataset. One can see, for example, that the *ln-cof* axis is horizontal in the respective view – hence the respective dimension varies strongest left-to-right in that scatterplot. This is confirmed by noting the dark blue to yellow-red color gradient observable along the same direction and noting that color maps the dimension *ln-cof*.

Biplot axes have several other uses such as finding dimensions which are correlated or independent of each other. While simple to compute and interpret, and applicable to both 2D and 3D projections,

they are *global* explanations of a projection. Indeed, if a projection exhibits *e.g.* multiple clusters and the direction of strongest variation of some variable differs among these clusters, then the respective biplot axis will show only the average direction among all clusters, which may not be insightful.

Axis legends: In contrast to the previous mechanisms which aimed to explain the scatterplot, axis legends aim to explain the meaning of the viewing space dimensions in terms of data dimensions. Introduced by Broeksema et al. (2013) for 2D linear projections, these work as follows. For each of the x and y screen axes, a bar chart of n bars is displayed, showing how much of the variance of the n data dimensions is captured by the respective screen axis. Intuitively put, compared to classical Cartesian scatterplots, where the x and y screen axes are mapped to a single data dimension, axis legends show that, in a projection, the screen axes encode a mix of different dimensions with different weights. Hence, long bars indicate data dimensions that explain well the spread of points along the respective screen axis, and conversely. Coimbra et al. (2016) extended this idea to 3D projections (linear or nonlinear) by adding a third legend for the depth axis. Long bars in this third legend indicate thus dimensions that one cannot see in the projection viewed from the current viewpoint. Figure 2.5 illustrates this. We see here that the dimension *ln-cof* has the largest bar in the x axis legend, which matches the fact that the biplot axis for that dimension is oriented horizontally.

Axis legends are slightly more complex to learn to use and interpret than color coding and biplot axes. Also, as biplot axes, these are global techniques as they explain the screen axes and not the individual visual patterns (*e.g.* clusters) in a projection.

Viewpoint legends and correlation views: Introduced by Coimbra et al. (2016), these tools jointly aim to explain which dimensions are best visible for a given viewpoint for a 3D projection. Figure 2.5 illustrates both techniques. The viewpoint legend shows a sphere on whose surface one conceptually maps all possible viewpoints one could look at the 3D projection from. For every such viewpoint, the technique computes the two data dimensions (of the total n ones) which are best examinable from the respective viewpoint, *i.e.*, whose bars would be the longest in the x and y axis legends for that viewpoint. The respective dimension-pair is then color-coded and mapped to the sphere surface. The color mapping is depicted separately in the correlation view, which is a matrix plot showing, for all dimension-pairs, which ones were chosen for color coding some viewpoints on the sphere. For example, in Fig. 2.5, the current viewpoint, depicted by the small crosshair in the middle of the viewpoint legend sphere, is surrounded by a large green region. Looking at the correlation view, we find that green encodes the dimension-pair (*ln-cof*, *ln-sum-tloc*), which matches the fact that, in the

current view, these dimensions have the longest bars in the x and y axis legends, respectively. Since the green region on the sphere is quite large, rotating the viewpoint from its current position will, for some amount of time, still show best the same dimension-pair in the projection.

Viewpoint legends and correlation views are specific explanatory tools for 3D projections. However, they are limited in scalability to datasets having about 10 dimensions (due to the use of the matrix plot to show dimension pairs). Also, they use, per viewpoint, only two of the n data dimensions to characterize (explain) the 3D projection viewed from that viewpoint.

Dimension color coding: The final explanatory technique for projections we discuss here takes a radically different approach from the above ones. Rather than explaining every point separately (as in color coding) or explaining the entire projection in terms of data dimensions (as the other techniques discussed above), dimension color coding, proposed originally by da Silva et al. (2015), aims to explain *groups* of *close* points in the projection. These coincide well with the aforementioned idea of explaining visual patterns in a projection. Indeed, such visual patterns are naturally formed by groups of close points. The technique works as follows. For every point $V(\mathbf{x}_i)$ in the projection, data points $\mathbf{x}_j$ in D which project close to $V(\mathbf{x}_i)$ are selected. The data dimension which varies least over the set of selected points is next determined and color-coded to mark $V(\mathbf{x}_i)$. The brightness of the used colors indicates how much of the data variance in the studied neighborhood is captured by the color-coded dimension.

Figure 2.6 illustrates this for the software dataset using a 2D projection constructed by LAMP – the same projection as in Fig. 2.4a. Points are drawn as disks so as to fill the gaps between them in the projection and create compact color patterns that are arguably easier to interpret than discrete colored scatterplots. The depicted projection has two 'lobes' which both contain a purple region, meaning that samples in such a region have similar values of the *lines of code* dimension. The fact that there are two such distinct regions is explained, upon further investigation, by the fact that the two regions refer to different *values* of the *lines of code* dimension. Since *lines of code* characterizes the size of a software system, note that this matches the explanatory labels (from Coimbra et al. (2016)) in Fig. 2.4a) which explain those two lobes as containing large, respectively small, software systems.

Dimension color coding has several attractive properties. It can explain local patterns in a projection, unlike any of the techniques discussed so far. These patterns do not need to be manually selected but rather are determined by the size of the considered neighborhoods in the computation of the explanation. The explanation is quite straightforward to interpret and does not take additional screen space unlike the axis legends, viewpoint legends and correlation views, or biplot axes.

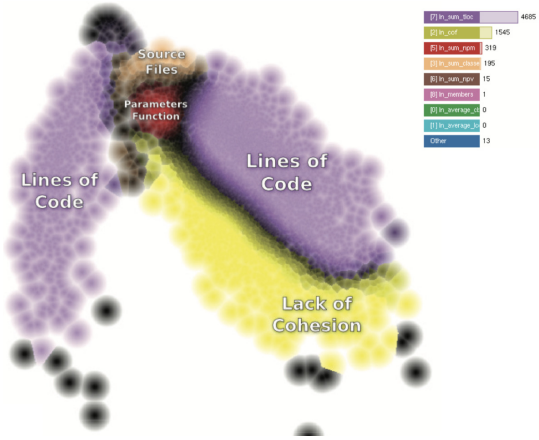


Figure 2.6: Dimension color coding explanation for a software dataset. Figure taken from da Silva et al. (2015).

However, it is limited by its use of categorical color coding to showing only about 8 to 10 dimensions. More importantly, it uses a single and simple criterion to determine what explains a neighborhood in a projection, namely the variance of dimensions over its points. Also, this technique has been applied so far only to explain 2D projections.

Summarizing our discussion of 2D and 3D projections, we find that both projection types are assisted in different ways by several existing explanatory techniques. However, no such technique manages to explain well all patterns visible in the projection in terms of data dimensions. Separately, 3D projections have a number of arguable advantages but also arguable limitations compared to 2D ones, and only limited studies exist that compare 2D and 3D techniques. 3D projections also strongly depend, in their understanding, on the viewpoint one chooses to examine them from. Finally, the comparison of 2D *vs* 3D techniques have been mostly limited to using raw, unannotated, projections for this task. Finding out how such projections fare with respect to each other when enhanced by explanatory techniques is still a largely open issue. We address several of these limitations further in our work. Specifically, in Chapter 5, we extend the dimension color coding explanation to use additional criteria involving the dataset. Separately, in Chapter 6, we apply such explanations to both 2D and 3D projections and compare them both quantitatively and qualitatively for a number of datasets and projection techniques.

3

INTERACTIVE AXIS-BASED 3D ROTATION SPECIFICATION USING IMAGE SKELETONS

Abstract: Specifying 3D rotations of shapes around arbitrary axes is not easy to do. We present a new method for this task, based on the concept of natural local rotation axes inferred from the local shape structure, in support of 3D exploration and manipulation tasks. We define such axes using the 3D curve skeleton of the shape of interest. We compute effective and efficient approximations of such skeletons using the 2D projection of the shape. Our method allows users to specify 3D rotations around parts of arbitrary 3D shapes with a single click or touch (plus a drag motion), is simple to implement, works in real time for large scenes, can be easily added to any OpenGL-based scene viewer, and can be used on both mouse-based and touch interfaces [1].

3.1 INTRODUCTION

Interactive manipulation of 3D scenes is a key part of many applications such as CAD/CAM modeling, computer games, and scientific visualization (Jackson et al., 2013). 3D rotations are an important manipulation type, as they allow examining scenes from various viewpoints to *e.g.* select the most suitable one for the task at hand. Two main 3D rotation types exist – rotation around a *center* and rotation around an *axis*. The first one can be easily specified via classical (mouse-and-keyboard) (Zhao et al., 2011) interfaces or touch interfaces (Yu et al., 2010) by well-known metaphors such as the trackball. The latter is also easy to specify if the rotation axis coincides with one of the world-coordinate axes. Rotations around arbitrary axes are considerably harder to specify, as this requires a total of 7 degrees of freedom (6 for specifying the axis and one for the rotation angle around the axis).

For certain tasks, users do not need to rotate around *any* 3D axis. Consider examining a (complex) 3D shape such as a statue: We can argue that a natural way to display this shape is with the statue's head upwards; and a good way to explore the shape from all viewpoints is to rotate it around its vertical axis while keeping its upwards orientation fixed. This keeps the shape's global orientation (which helps understanding the shape) but allows one to examine it from all viewpoints

Several methods support the above exploration scenario by first aligning a shape's main symmetry axis with one of the world coordinate axes and then using a simple-to-specify rotation around this world

1 This chapter is based on the paper 'Interactive Axis-Based 3D Rotation Specification Using Image Skeletons' Zhai et al. (2020).

axis (Duffin and Barrett, 1994). This scenario falls short when (a) the studied shape does not admit a *global* symmetry axis, although its parts may have local symmetry axes; (b) computing such (local or global) symmetry axes is not simple; or (c) we do not want to rotate along an axis which is first aligned with a world axis.

To address the above, we propose a novel interaction mechanism: Given a shape viewed from an arbitrary 3D viewpoint, we allow the user to choose a part of interest of the shape. Next, we propose a fast and generic method to compute an approximate 3D symmetry axis for this part. Finally, we interactively rotate the shape around this axis by the desired angle. This effectively allows one to rotate the viewpoint to examine shapes around a multitude of symmetry axes that the users can easily select. Our method can handle any 3D shape or scene, *e.g.*, polygon mesh or polygon soup, point-based or splat-based rendering, or combination thereof; is simple to implement and works at interactive rates even for scenes of hundreds of thousands of primitives; requires no preprocessing of the 3D geometry; and, most importantly, allows specifying the rotation axis and rotation angle by a single click (followed by a drag motion), therefore being suitable for both classical (mouse-based) and touch interfaces. We demonstrate our method on several types of 3D scenes.

3.2 RELATED WORK

We structure related work around two topics, namely methods for specifying rotations of 3D objects around arbitrary axes (Sec. 3.2.1) and methods for computing medial descriptors that capture a shape's symmetry (Sec. 3.2.2).

3.2.1 *Rotation specification*

3D rotations can be specified by many techniques. The trackball metaphor (Chen et al., 1998) is one of the oldest and likely most popular techniques. Given a 3D center-of-rotation $\mathbf{x}$, the scene is rotated around an axis passing through $\mathbf{x}$ and determined by the projections on a hemisphere centered at $\mathbf{x}$ of the 2D screen-space locations $\mathbf{p}_1$ and $\mathbf{p}_2$ corresponding to a (mouse) pointer motion. The rotation angle α is controlled by the amount of pointer motion. While simple to implement and use, trackball rotation does not allow precise control of the actual axis around which one rotates, as this axis constantly changes while the user moves the pointer (Bade et al., 2005; Zhao et al., 2011). Several usability studies of trackball and alternative 3D rotation mechanisms explain these limitations in detail (Jacob and Oliver, 1995; Hinckley et al., 1997; Frokjaer et al., 2000; Partala, 1999). Several refinements of the original trackball (Chen et al., 1998) were proposed to address

these (Hultquist, 1990; Shoemake, 1992). In particular, Henriksen et al. (2004a) formally analyze the trackball's principle and its limitations and also propose improvements which address some, but not all, limitations. At the other extreme, world-coordinate-axis rotations allow rotating a 3D scene around the x, y, or z axes (Zhao et al., 2011; Jackson et al., 2013). The rotation axis and rotation angle are chosen by simple click-and-drag gestures in the viewport. This works best when the scene is already *pre-aligned* with a world axis, so that rotating around that axis yields meaningful viewpoints.

Pre-alignment of 3D models is a common preprocessing stage in visualization (Chaouch and Verroust-Blondet, 2009). Principal Component Analysis (PCA) does this by computing a 3D shape's eigenvectors $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$, ordered by their eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$, so that the coordinate system $\{\mathbf{e}_i\}$ is right-handed. Next, the shape is aligned with the viewing coordinate system (x, y, z) by a simple 3D rotation around the shape's barycenter (Tangelder and Veltkamp, 2008; Kaye and Ivrissimtzis, 2015). Yet, pre-alignment is not effective when the scene does not have a clear main axis (λ_1 close to λ_2) or when the major eigenvector does not match the rotation axis desired by the user.

3D rotations can be specified by classical (mouse-and-keyboard) mechanisms (Zhao et al., 2011) but also touch interfaces. Yu et al. (2010) present a direct-touch exploration technique for 3D scenes called Frame Interaction with 3D space (FI3D). Guo et al. (2017) extend FI3D with constrained rotation, trackball rotation, and rotation around a user-defined center. Yu and Isenberg (2009) used trackball interaction to control rotation around two world axes by mapping it to single-touch interaction. Hancock et al. (2007, 2010) use two or three touch input to manipulate 3D shapes on touch tables and, in this context, highlighted the challenge of specifying 3D rotations. All above works stress the need for *simple* rotation-specification mechanisms using a minimal number of touch points and/or keyboard controls.

3.2.2 *Medial descriptors*

Medial descriptors, also known as skeletons, have been used for decades to capture the symmetry structure of shapes (Blum, 1967; Siddiqi and Pizer, 2008). For shapes $\Omega \subset \mathbb{R}^n$, $n \in \{2, 3\}$ with boundary $\partial\Omega$, skeletons are defined as

$$S_\Omega = \{\mathbf{x} \in \Omega) | \exists \mathbf{f}_1 \in \partial\Omega, \mathbf{f}_2 \in \partial\Omega : \mathbf{f}_1 \neq \mathbf{f}_2 \wedge ||\mathbf{x}-\mathbf{f}_1|| = ||\mathbf{x}-\mathbf{f}_2|| = DT_\Omega(\mathbf{x}\} \tag{3.1}$$

where $\mathbf{f}_i$ are called the *feature points* (Meijster et al., 2002) of skeletal point $\mathbf{x}$ and DT_Ω is the distance transform (Rosenfeld and Pfaltz, 1968; Costa and Cesar, 2000) of skeletal point $\mathbf{x}$, defined as

$$DT_\Omega(\mathbf{x} \in \Omega) = \min_{\mathbf{y} \in \partial\Omega} \|\mathbf{x} - \mathbf{y}\| \tag{3.2}$$

These feature points define the so-called *feature transform* (Hesselink and Roerdink, 2008; Tagliasacchi et al., 2016)

$$FT_\Omega(\mathbf{x} \in \Omega) = \arg\min_{\mathbf{y} \in \partial\Omega} \|\mathbf{x} - \mathbf{y}\|, \tag{3.3}$$

which gives, for each point $\mathbf{x}$ in a shape Ω, its set of feature points on $\partial\Omega$, or contact points with $\partial\Omega$ of the maximally inscribed disk in Ω centered at $\mathbf{x}$.

Many methods compute skeletons of 2D shapes, described as either polyline contours (Ogniewicz and Kubler, 1995) or binary images (Telea and van Wijk, 2002; Costa and Cesar, 2000; Falcão et al., 2004; Falcao et al., 2017). State-of-the-art methods *regularize* the skeleton by removing its so-called spurious branches caused by small noise perturbations of the boundary $\partial\Omega$, which bring no added value, but only complicate further usage of the skeleton. Regularization typically defines a so-called *importance* $\rho(\mathbf{x}) \in \mathbb{R}^+ | \mathbf{x} \in S_\Omega$ which is low on noise branches and high elsewhere on S_Ω. Several authors (Falcão et al., 2004; Ogniewicz and Kubler, 1995; Costa and Cesar, 2000; Telea and van Wijk, 2002; Falcao et al., 2017) set ρ to the length of the shortest path along $\partial\Omega$ between the two feature points $\mathbf{f}_1$ and $\mathbf{f}_2$ of $\mathbf{x}$. If more than two such feature points exist, existing methods typically just choose two points from the available ones. Upper thresholding ρ by a sufficiently high value removes noise branches. Importance regularization can be efficiently implemented on the GPU (Ersoy et al., 2011) using fast distance transform computation (Cao et al., 2010). Overall, 2D skeletonization can be seen, from a practical perspective, as a solved problem.

In 3D, two main skeleton types exist (Tagliasacchi et al., 2016): *Surface skeletons*, defined by Eqn 3.1 for $\Omega \subset \mathbb{R}^3$, consist of complex intersecting manifolds with boundary, and hence are hard to compute and utilize (Tagliasacchi et al., 2016). *Curve skeletons* are curve-sets in $\mathbb{R}^3$ that locally capture the tubular symmetry of shapes (Cornea et al., 2007). They are structurally much simpler than surface skeletons and enable many applications such as shape segmentation (Rodrigues et al., 2018) and animation (Bian et al., 2018). Yet, they still cannot be computed in real time, and require a well-cured definition of Ω as a watertight, non-self-intersecting, fine mesh (Sobiecki et al., 2013) or a high-resolution voxel volume (Reniers et al., 2008; Falcao et al., 2017).

Kustra et al. (2013) and Livesu et al. (2012) address the above challenges of 3D curve-skeleton computation by using an *image-based* ap-

proach. They compute an approximate 3D curve skeleton from 2D skeletons extracted from multiple 2D views of a shape. While far simpler and also more robust than true 3D skeleton extraction, such methods need hundreds of views and cannot be run at interactive rates. Our proposal also uses an image-space skeleton computation, but uses different, simpler, heuristics than Kustra et al. (2013); Livesu et al. (2012) to estimate 3D depth, and a single view, thereby achieving the speed required for interactivity.

3.3 PROPOSED METHOD

We construct a 3D rotation in five steps (Fig. 3.1). We start by loading the scene of interest – any arbitrary collection of 3D primitives, with no constraints on topology or sampling resolution – into the viewer (a). Next, the user can employ any mechanisms offered by the viewer, *e.g.* trackball rotation, zoom, or pan, to choose a *viewpoint of interest*, from which the scene shows a detail around which one would like to further rotate to explore the scene. In our example, such a viewpoint (b) shows the horse's rump, around which – for the sake of illustration – we want to rotate to examine the horse from different angles (Fig. 3.1).

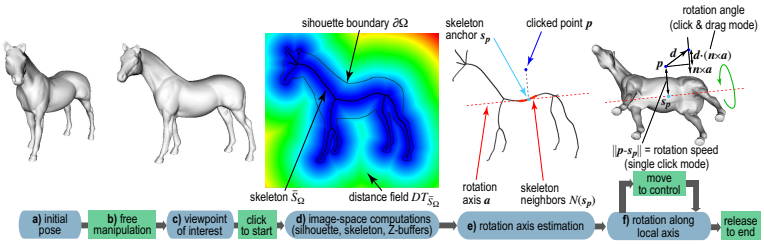


Figure 3.1: Skeleton-based local rotation pipeline. Blue boxes indicate tool states. Green boxes indicates user actions.

3.3.1 *Rotation axis computation*

From the above-mentioned initial viewpoint, we next perform three image-space operations to compute the 3D rotation axis. These steps, denoted A, B, and C next, are as follows.

A. Silhouette extraction: This is the first operation in Fig. 3.1, step (d). We render the shape with Z-buffering on and using the *GL_LESS* OpenGL depth-test. Let Ω_{near} be the resulting Z-buffer. We next find the silhouette Ω of the shape as all pixels that have a value in Ω_{near} different from the default (the latter being 1 for standard OpenGL settings).

B. Skeleton computation: We next compute the silhouette skeleton S_Ω (Eqn. 3.1) by the method in Telea and van Wijk (2002) (Fig. 3.1, step (d)). To eliminate spurious skeletal branches caused by small-scale noise along $\partial\Omega$, we regularize S_Ω by the salience-based metric in Telea (2011). This regularization works as follows – see also the sketch in Fig. 3.2c. For every point $\mathbf{x} \in S_\Omega$ of the full skeleton delivered by Eqn. 3.1, we first compute the importance ρ (Telea and van Wijk, 2002), *i.e.*, the shortest path along $\partial\Omega$ between the two feature points of $\mathbf{x}$ (see also Sec. 3.2). This path is marked red in Fig. 3.2c. As shown in Telea and van Wijk (2002); Falcao et al. (2017); Tagliasacchi et al. (2016), and also outlined in Sec. 3.2, ρ monotonically increases along skeletal branches from their endpoints to the skeleton center, and equals, for a skeleton point $\mathbf{x}$, the amount of boundary which is captured (described) by $\mathbf{x}$.

We next define the *salience* of skeletal point $\mathbf{x}$ as

$$\sigma(\mathbf{x}) = \frac{\rho(\mathbf{x})}{DT_\Omega(\mathbf{x})}, \tag{3.4}$$

that is, the importance ρ normalized by the skeletal point's distance to boundary. As shown in Telea (2011), σ is *overall high* on skeleton branches caused by important (salient) cusps of $\partial\Omega$ and *overall low* on skeleton branches caused by small-scale details (noise cusps) along $\partial\Omega$. Figure 3.2c shows this for a small cusp on the boundary of a 2D silhouette of a noisy 3D dino shape. As we advance in this image along the black skeleton branch into the shape's rump (going below the grey area in the picture), ρ stays constant, but the distance to boundary DT_Ω increases, causing σ to decrease. Hence, we can regularize S_Ω simply by removing all its pixels having a salience value lower than a fixed threshold σ_0. Following Telea (2011), we set $\sigma_0 = 1$. Fig. 3.2 illustrates this regularization by showing the raw skeleton S_Ω and its regularized version

$$\overline{S}_\Omega = \{\mathbf{x} \in S_\Omega | \sigma(\mathbf{x}) \geq \sigma_0\} \tag{3.5}$$

for the noisy dino shape. Salience regularization (Fig. 3.2b) removes all spurious branches created by boundary noise, but leaves the main skeleton branches, corresponding to the animal's limbs, rump, and tail, intact. Images (d-g) in the figure show the silhouette Ω, importance ρ, distance transform DT_Ω, and salience σ for a zoom-in area around the shape's head, for better insight. Looking carefully at image (e), we see that ρ has non-zero values also outside the main skeleton branch corresponding to the animal's neck, visible as light-blue pixels. While such details may look insignificant, they are crucial: Thresholding ρ by too low values – the alternative regularization to our proposal – keeps many spurious skeletal branches, see the red inset in Fig. 3.2a. In contrast, σ is practically zero outside the neck branch (Fig. 3.2g). So, thresholding σ by $\sigma_0 = 1$ yields a clean skeleton, see the red inset

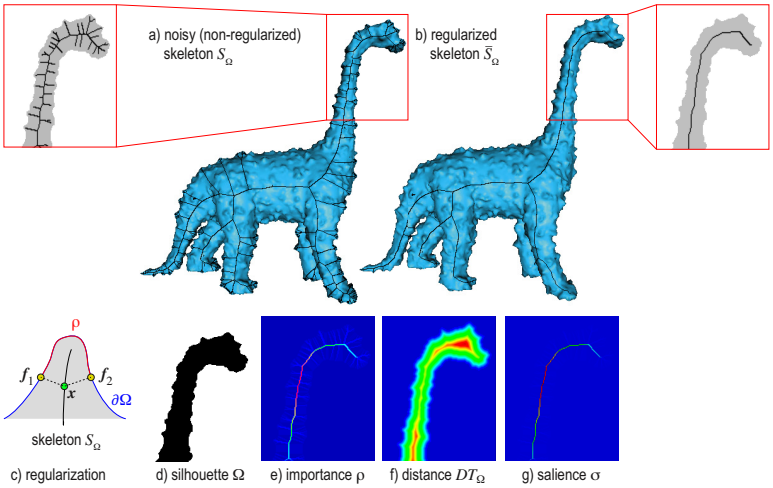


Figure 3.2: Raw skeleton S_Ω with (a) noise-induced branches and (b) salience-based regularized skeleton $\overline{S}_\Omega$. c) Principle of salience regularization. (d-g) Details of silhouette, importance, distance transform, and salience values for the noisy dino's head.

in Fig. 3.2b. Salience regularization is simple and automatic to use, requiring no free parameters, and hence preferable to ρ regularization – which requires careful setting of the threshold for ρ – or to any other skeleton regularization we are aware of. For further details on salience regularization, we refer to Telea (2011) and also its public implementation (Telea, 2014a).

C. Rotation axis computation: This is step (e) in Fig. 3.1. Let $\mathbf{p}$ be the pixel under the user-controlled pointer (blue in Fig. 3.1e). We first find the closest skeleton point $\mathbf{s}_p = \arg\min_{\mathbf{y}\in\overline{S}_\Omega} \|\mathbf{p}-\mathbf{y}\|$ by evaluating the feature transform (Eqn. 3.3) $FT_{\overline{S}_\Omega}(\mathbf{p})$ of the regularized skeleton $\overline{S}_\Omega$ at $\mathbf{p}$. Figure 3.1d shows the related distance transform $DT_{\overline{S}_\Omega}$. In our case, $\mathbf{s}_p$ is a point on the horse's rump skeleton (cyan in Fig. 3.1e). Next, we find the neighbor points $N(\mathbf{s}_p)$ of $\mathbf{s}_p$ by searching depth-first from $\mathbf{s}_p$ along the pixel connectivity-graph of $\overline{S}_\Omega$ up to a fixed maximal distance set to 10% of the viewport size. $N(\mathbf{s}_p)$ contains skeletal points along a single branch in $\overline{S}_\Omega$, or a few connected branches, if $\mathbf{s}_p$ is close to a skeleton junction. In our case, $N(\mathbf{s}_p)$ contains a fragment of the horse's rump skeleton (red in Fig. 3.1e). For each $\mathbf{q} \in N(\mathbf{s}_p)$, we set the depth $\mathbf{q}_z$ as the average of $\Omega_{far}(\mathbf{q})$ and $\Omega_{near}(\mathbf{q})$. Here, Ω_{near} is the Z-buffer of the scene rendered as described in step A above; and Ω_{far} is the Z-buffer of the scene rendered as before, but with front-face culling on, *i.e.*, the depth of the *nearest* backfacing polygons to the view plane.

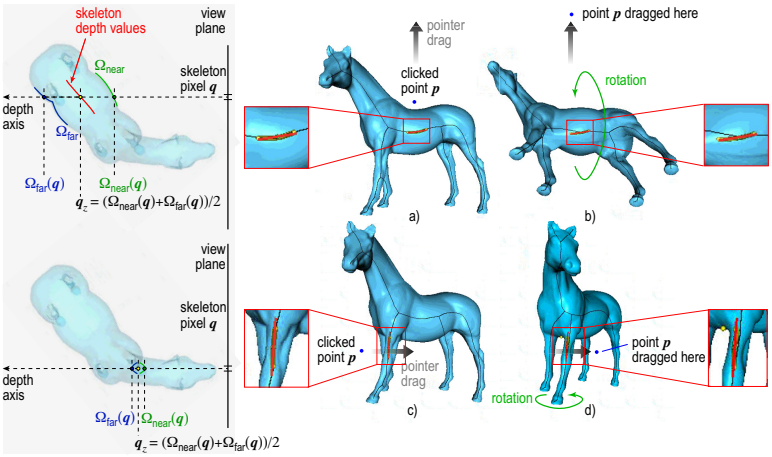


Figure 3.3: Depth estimation of rotation axis for (a,b) non-overlapping part and (c,d) overlapping parts. In both cases, the rotation axis (red) is nicely centered in the shape. See Sec. 3.3.1.

Fig. 3.3 shows how this works. The user clicks above the horse's rump and drags the pointer upwards (a). Image (b) shows the resulting rotation. As visible in the inset in (a), the rotation axis (red) is centered inside the rump, as its depth $\mathbf{q}_z$ is the average of the near and far rump faces. To better understand this, the image left to Fig. 3.3a shows the horse rendered transparently, seen from above. The depth values in Ω_{near} and Ω_{far} are shown in green, respectively blue. The skeleton depth values (red) are the average of these. Note that, when the rotation ends, the new silhouette skeleton does *not* match the rotation axis – see inset in (b). This is normal and expected. If the user wants to start a new rotation from (b), then the 2D skeleton from this image will be used to compute a new, matching, rotation axis.

Next, we consider a case of overlapping shape parts (Fig. 3.3c). The user clicks left to the horse's left-front leg, which overlaps the right-front one, and drags the pointer to the right. Image (d) shows the resulting rotation. The rotation axis (red) is centered inside the left-front leg. In this case, $\Omega_{far}(\mathbf{q})$ contains the Z values of the backfacing part of the left-front leg, so $(\Omega_{near}(\mathbf{q}) + \Omega_{far}(\mathbf{q}))/2$ yields a value roughly halfway this leg along the Z axis. The image left to Fig. 3.3c clarifies this by showing the horse from above and the respective depth values in Ω_{near} (green) and Ω_{far} (blue).

Separately, we handle non-watertight surfaces as follows: If $\Omega_{far}(\mathbf{q})$ contains the default Z value (one), this means there's no backfacing surface under a given pixel $\mathbf{q}$, so the scene is not watertight at $\mathbf{q}$. We then set $\mathbf{q}_z$ to $\Omega_{near}(\mathbf{q})$.

We now have a set $N_{3D} = \{(\mathbf{q} \in N(\mathbf{s}_p), \mathbf{q}_z)\}$ of 3D points that approximate the 3D curve skeleton of our shape close to the pointer location

p. We set the 3D rotation axis **a** to the line passing through the average point of N_{3D} and oriented along the largest eigenvector of N_{3D}'s covariance matrix (Fig. 3.1e, red dotted line).

3.3.2 *Controlling the rotation*

We propose three interactive mechanisms to control the rotation (Fig. 3.1), step (f)):

- **Indication:** As the user moves the pointer **p**, we continuously update the display of **a**. This shows along which axis the scene *would* rotate if the user initiated rotation from **p**. If **a** is found suitable, one can start rotating by a click following one of the two modes listed next; else one can move the pointer **p** to find a more suitable axis;
- **Single click:** In this mode, we compute a rotation speed σ equal to the distance $\|\mathbf{p} - \mathbf{s}_p\|$ and a rotation direction δ (clockwise or anticlockwise) given by the sign of the cross-product $(\mathbf{s}_p - \mathbf{p}) \times \mathbf{n}$, where **n** is the viewplane normal. We next continuously rotate (spin) the shape around a with the speed σ in direction δ;
- **Click and drag:** Let **d** be the drag vector created by the user as she moves the pointer **p** from the current to the next place in the viewport with the control, *e.g.* mouse button, pressed. We rotate the scene around **a** with an angle equal to $\mathbf{d} \cdot (\mathbf{n} \times \mathbf{a})$ (Fig. 3.1e).

We stop rotation when the user releases the control (mouse button). In single-click mode, clicking closer to the shape rotates slowly, allowing to examine the shape in detail. Clicking farther rotates quicker to *e.g.* explore the shape from the opposite side. The rotation direction is given by the *side* of the skeleton where we click: To change from clockwise to counterclockwise rotation in Fig. 3.1, we only need to click below, rather than above, the horse's rump. In click-and-drag mode, the rotation speed and direction is given by the drag vector **d**: Values **d** orthogonal to the rotation axis **a** create corresponding rotations clockwise or anticlockwise around **a**; values **d** along **a** yield no rotation. This matches the intuition that, to rotate along an axis, we need to move the pointer *across* that axis.

The skeleton-based construction of the rotation axis is key to the effectiveness of our approach: If the shape exhibits some elongated structure in the current view (*e.g.* rump or legs in Fig. 3.1c), this structure will yield a skeleton branch. Clicking closer to this structure than to other structures in the same view – *e.g.*, clicking closer to the rump than to the horse's legs or neck – selects the respective skeleton branch to rotate around. This way, the 3D rotation uses the 'natural' structure of the viewed shape. We argue that this makes sense in an exploratory

scenario, since, during rotation, the shape parts we rotate around stay *fixed* in the view, as if one 'turns around' them.

The entire method requires a *single click* and, optionally, a pointer drag motion to execute. This makes our method simpler than other 3D rotation methods that rotate around freely specifiable 3D axes, and also directly applicable to contexts where no second button or modifier keys are available, *e.g.*, touch screens. Moreover, our method does not require any complex (and/or slow) 3D curve-skeleton computation: We compute only 2D (silhouette) skeletons, which are fast and robust to extract (Telea and van Wijk, 2002; Ersoy et al., 2011). We can handle any 3D input geometry, *e.g.*, meshes, polygon soups, point clouds, or mixes thereof, as long as such primitives render in the Z-buffer (see Sec. 3.4 for examples hereof).

3.3.3 *Improvements of the basic method*

We next present three improvements of the local-axis rotation mechanism described above.

Zoom level: A first issue regards computing the scene's 2D silhouette Ω (Sec. 3.3.1A). For this to work correctly, the entire scene must be visible in the current viewport. If this is not the case, the silhouette boundary $\partial\Omega$ will contain parts of the viewport borders. Fig. 3.4a shows this for a zoomed-in view of the horse model, with the above-mentioned border parts marked purple. This leads to branches in the skeleton S_Ω that do not provide meaningful rotation axes. We prevent this to occur by requiring that the entire scene is visible in the viewport before initiating the rotation-axis computation. If this is not the case, we do not allow the skeleton-based rotation to proceed, but map the user's interaction to standard trackball-based rotation.

Skeleton junctions: If the user selects $\mathbf{p}$ so that the skeleton anchor $\mathbf{s}_p$ is too close to a skeleton junction, then the neighbor-set $N(\mathbf{s}_p)$ will contain points belonging to more than two branches. Estimating a line from such a point set (Sec. 3.3.1C) is unreliable, leading to possibly meaningless rotation axes. Figures 3.4b-d illustrates the problem. The corresponding skeleton points $N(\mathbf{s}_p)$ used to estimate the axis are shown in yellow, and the resulting axes in red. When $\mathbf{s}_p$ is far from the junction (Figs. 3.4b,d), $N(\mathbf{s}_p)$ contains mainly points from a *single* skeleton branch, so the estimated rotation axes are reliable. When $\mathbf{s}_p$ is very close to a junction (Fig. 3.4c), $N(\mathbf{s}_p)$ contains points from all three meeting skeletal branches, so, as the user moves the pointer $\mathbf{p}$, the estimated axis 'flips' abruptly and can even assume orientations that do not match any skeleton branch.

We measure the *reliability* of the axis $\mathbf{a}$ by the anisotropy ratio $\gamma = \lambda_1/\lambda_3$ of the largest to smallest eigenvalue of N_{3D}'s covariance

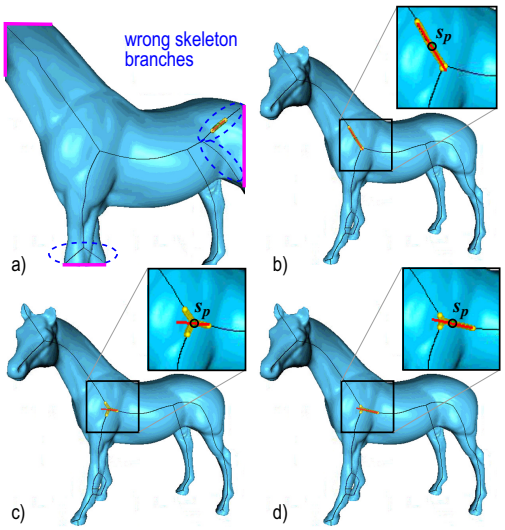


Figure 3.4: Two problems of estimating rotation axes from skeletons. (a) Zoomed-in scene. Anchor points close to (c), respectively farther from (b,d) a skeleton junction. See Sec. 3.3.3.

matrix. Other anisotropy metrics can be used equally well (Emory and Iaccarino, 2014). High γ values indicate elongated structures N_{3D}, from which we can reliably compute rotation axes. Low values, empirically detected as $\gamma < 5$, indicate problems to find a reliable rotation axis. When this occurs, we prevent executing the axis-based rotation.

Selection distance: A third issue concerns the position of the point $\mathbf{p}$ that starts the rotation: If one clicks too far from the silhouette Ω, the rotation axis $\mathbf{a}$ may not match what one expects. To address this, we forbid the rotation when the distance d from $\mathbf{p}$ to Ω exceeds a given upper limit d_{max}. That is, if the user clicks too far from any silhouette in the viewport, the rotation mechanism does not start. This signals to the user that, to initiate the rotation, she needs to click closer to a silhouette. We compute d as $DT_{\overline{\Omega}}(\mathbf{p})$, where $\overline{\Omega}$ is the viewpoint area outside Ω, *i.e.*, all viewport pixels where Ω_{near} equals the default Z-buffer value (see Sec. 3.3.1A).

We studied two methods for estimating d_{max} (see Fig. 3.5). First, we set d_{max} to a fixed value, in practice 10% of the viewport size. Using a constant d_{max} is however not optimal: We found that, when we want to rotate around *thick* shape parts, *e.g.* the horse's rump in Fig. 3.5b, it is intuitive to select $\mathbf{p}$ even quite far away from the silhouette. This is the case of point $\mathbf{p}_1$ in Fig. 3.5b. In contrast, when we want to rotate around *thin* parts, such as the horse's legs, it is not intuitive to initiate the rotation by clicking too far away from these parts. This is the situation of

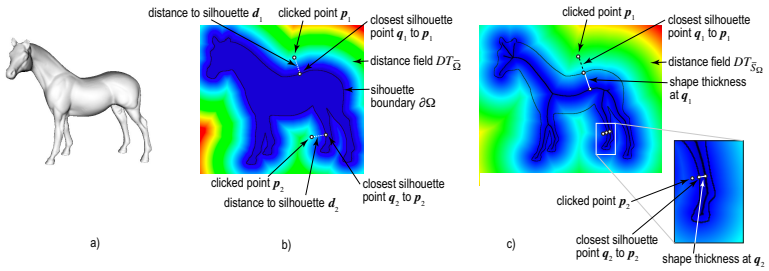


Figure 3.5: Improvements of axis-based rotation method. (a) A view of the shape to be rotated. (b) Fixed maximum-distance setting for two clicked points $\mathbf{p}_1$ and $\mathbf{p}_2$. (c) Thickness-based maximum-distance setting for two clicked points $\mathbf{p}_1$ and $\mathbf{p}_2$.

point $\mathbf{p}_2$ in Fig. 3.5b. Hence, d_{max} depends on the scale of the shape part we want to rotate around; selecting large parts can be done by clicking farther away from them than selecting small parts.

We model this by setting d_{max} to the local *shape thickness* (Fig. 3.5c). We estimate thickness as follows: We find the closest point on the silhouette boundary $\partial\Omega$ to the clicked point $\mathbf{p}$ as $\mathbf{q} = FT_{\overline{\Omega}}(\mathbf{p})$. The shape thickness at $\mathbf{q}$ is the distance to the skeleton, *i.e.*, $DT_{\overline{S}_\Omega}(\mathbf{q})$. This is the 2D equivalent of the more general 3D-shape-thickness estimation proposed in Telea and Jalba (2011). In Fig. 3.5c, the point $\mathbf{p}_1$ is the farthest clickable point around $\mathbf{q}_1$ to the silhouette that allows starting a rotation around the rump. If we click further from the silhouette than the distance d_{max} from $\mathbf{p}_1$ to $\mathbf{q}_1$, no rotation is done. For the leg part, the farthest clickable point around $\mathbf{q}_2$ must, however, be much closer to the silhouette (Fig. 3.5c), since here the local shape thickness (distance d_{max} from $\mathbf{p}_2$ to $\mathbf{q}_2$) is smaller.

3.4 RESULTS

Figure 3.6 shows our 3D skeleton-based rotation applied to two 3D mesh models – a hand and a ship. For extra insights, we recommend also watching the demonstration videos (Zhai et al., 2019). First, we consider a 3D mesh model of a human hand (100K faces), which is not watertight (open at wrist). We start from a poor viewpoint from which we cannot easily examine the shape (a). We click close to the thumb (b) and drag to rotate around it (b-e), yielding a better viewpoint (e). Next, we want to rotate around the shape to see the other face, but keeping the shape roughly in place. Using a trackball or world-coordinate axis rotation cannot easily achieve this. We click on a point close to the shape-part we want to keep *fixed* during rotation (f), near the the wrist, and start rotation. Images (g-j) show the resulting rotation.

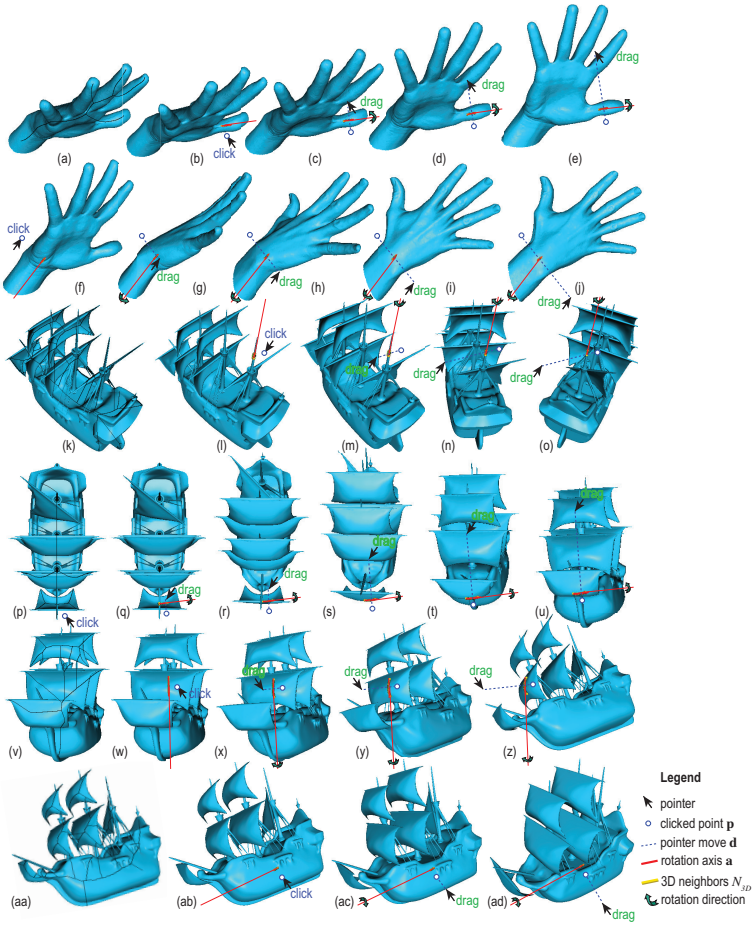


Figure 3.6: Examples of two rotations (a-e), (f-j) for the hand shape and four rotations (k-o), (p-u), (v-z), (aa-ad) for the ship.

Figure 3.6(k-ad) show a more complex ship object (380K polygons). This mesh contains multiple self-intersecting and/or disconnected parts, some very thin (sails, mast, ropes) (Kustra et al., 2014). Computing a 3D skeleton for this shape is extremely hard or even impossible, as Eqn. 3.1 requires a watertight, non-self-intersecting, connected shape boundary $\partial\Omega$. Our method does not suffer from this, since we compute the skeleton of the *2D silhouette* of the shape. We start again from a poor viewing angle (k). Next, we click close to the back mast to rotate around it, showing the ship from various angles (l-o). Images (p-u) show a different rotation, this time around an axis found by clicking close to the front sail, which allows us to see the ship from the front. Note how the 2D skeleton has changed after this rotation – compare images (p) with (v). This allows us to select a new rotation axis by clicking on the main sail, to see

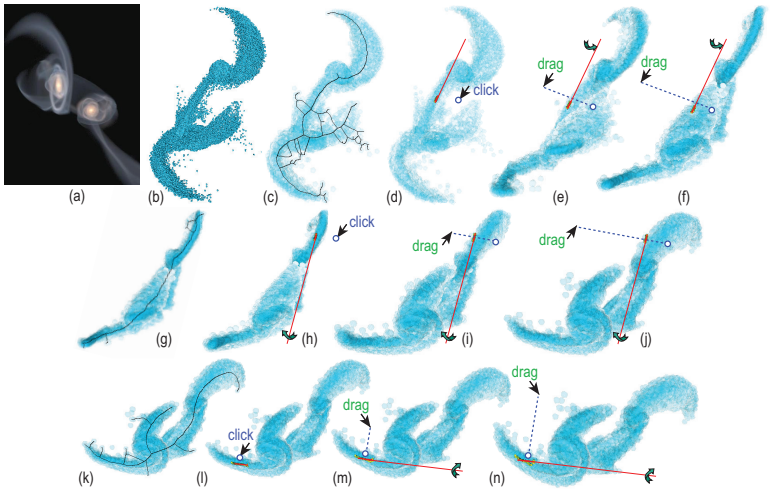


Figure 3.7: Exploration of astronomical point cloud dataset. (a) Volume-rendered overview J. Dubinski *et al.* (2006). Rotations around three 3D axes (b-f), (g-j), (k-n).

the ship's stern from below (w-z). Finally, we click on the ship's rump (aa) to rotate the ship and make it vertical (ab-ad). The entire process of three rotations took around 20 seconds.

Figure 3.7 shows a different dataset type – a 3D point cloud that models a collision simulation between the Milky Way and the nearby Andromeda Galaxy (Dubinski, 2001; J. Dubinski *et al.*, 2006). Its 160K points describe positions of the stars and dark matter in the simulation. Image (a) uses volume rendering to show the complex structure of the cloud, for illustration purposes – we do not use this rendering in our method. Rather, we render the cloud in our pipeline using 3D spherical splats (b). Image (c) shows the cloud, rendered with half-transparent splats, so that opacity reflects local point density. Since we render a 3D sphere around each point, this results in a front and back buffer Ω_{near} and Ω_{far}, just as when rendering a 3D polygonal model. From these, we can compute the 2D skeleton of the cloud's silhouette, as shown in the figure. Images (d-f) show a rotation around the central tubular structure of the cloud, which reveals that the cloud is relatively flat when seen from the last viewpoint (f). Image (g) shows the new 2D skeleton corresponding to the viewpoint after this rotation. We next click close to the upper high-density structure (f) and rotate around it. Images (h-j) reveal a spiral-like structure present in the lower part of the cloud, which was not visible earlier. To explore this structure better, we next click on its local symmetry axis (l) and rotate around it. Images (l-n) reveal now better this structure. As for the earlier examples, executing these three rotations took roughly 15 seconds. Scientists involved

with studying this dataset for roughly a decade appreciated positively the ease of use of the skeleton-based rotation as compared to standard trackball and multi-touch gestures.

3.5 DISCUSSION

We next outline our method's advantages and limitations:

Ease of application: We can rotate around 3D axes locally aligned with the scene's features with a single click and optionally pointer drag motion. This makes our method usable to contexts where no second button, modifier keys, or multi-touch input is available. Finding the axis works with even inexact click locations as we use a *set* of closest 2D-skeleton points for that ($N(\mathbf{s}_p)$, Sec. 3.3).

Reversibility: Since 3D rotation axes are computed from 2D silhouette skeletons, rotations are not, strictly speaking, invertible: Rotating from a viewpoint $\mathbf{v}_1$ with an angle α around a 3D local axis $\mathbf{a}_1$ computed from the silhouette Ω_1 leads to a viewpoint $\mathbf{v}_2$ in which, from the corresponding silhouette Ω_2, a different axis $\mathbf{a}_2 \neq \mathbf{a}_1$ can be computed. This is however a problem only if the user *releases* the pointer (mouse) button to end the rotation; if the button is not released, the computation of a new axis $\mathbf{a}_2$ is not started, so moving the pointer back will reverse the rotation.

Genericity: We handle 3D meshes, polygon soups, and point clouds; our only requirement is that these generate fragments with a depth value. This contrasts using 3D curve skeletons for interaction, which heavily constrain the input scene quality, and cannot be computed in real time, as already mentioned. Also, the skeleton tool can be directly combined (used alongside) any other interaction tool, such as trackball, with no constraints.

Novelty: To our knowledge, this is the first time that 2D image-based skeletons have been used to perform interactive manipulations of 3D shapes. Compared to similar view-based reconstructions of 3D curve skeletons from their 2D silhouettes (Livesu et al., 2012; Kustra et al., 2013), our method requires a *single* viewpoint to compute an approximate 3D curve skeleton and is two to three orders of magnitude faster.

Scalability and implementation simplicity: Our method uses OpenGL 1.1 (primitive rendering and Z-buffer reading) plus the 2D image-based skeletonization method in Telea and van Wijk (2002) used to compute the skeleton S_Ω, its regularization $\overline{S}_\Omega$, and the feature transform $FT_{\overline{S}_\Omega}$. We implemented skeletonization in NVidia's CUDA and

C++ to handle scenes of hundreds of thousands of polygons rendered at 1000^2 pixel resolution in a few milliseconds on a consumer-grade GPU, *e.g.* GTX 660. The skeletonization computational complexity is linear in the number of silhouette pixels, *i.e.*, $O(|\Omega|)$. This is due to the fact that the underlying distance transform used has the same linear complexity. For details on this, we refer to the original algorithm (Cao et al., 2010). The separate code of this skeletonization method is available at A. Telea (2019).

Implementing the two improvements presented in Sec. 3.3 is also computationally efficient: The skeleton's distance transform $DT_{\overline{S}_\Omega}$ is already computed during the rotation axis estimation (Sec. 3.3.1C). The distance $DT_{\overline{\Omega}}$ and feature transforms $FT_{\overline{\Omega}}$ require one extra skeletonization pass of the background image $\overline{\Omega}$. All in all, our interaction method delivers frame rates over 100 frames-per-second on the aforementioned consumer-grade GPU. For replication purposes, the full code of the method is provided online (Zhai et al., 2019).

Limitations: An important limitation regards the effectiveness of our rotation mechanism. While our tests show that one can easily rotate a scene around its parts, it is still unclear which specific *tasks* are best supported by this rotation, and by how much so, as compared to other rotation mechanisms such as trackball. In the next chapter, we will aim to quantify this aspect by presenting several controlled user experiments in which we select a specific task to be completed with the aid of rotation and quantitatively compare (evaluate) the effectiveness of our rotation mechanism as compared to another established mechanisms, the virtual trackball.

Another more subtle point of discussion concerns the definition of choosing a good *viewpoint*. Technically speaking, there is no difference, in terms of implementation, between rotating a shape along some rotation axis (and keeping the OpenGL viewpoint fixed) and changing the viewpoint (but keeping the shape fixed), as both end up to what is known as the OpenGL *modelview* transform. However, users may perceive the two tasks – rotating a shape while keeping the viewpoint fixed *vs* changing the viewpoint but keeping a shape fixed, *i.e.*, looking at a fixed shape from different viewpoints – as different. Exploring how actual users form a mental model of the two types of operations and whether they see significant differences between the two is an interesting and important future work direction.

3.6 CONCLUSION

We proposed a novel method for specifying interactive rotations of 3D scenes around local axes using image skeletons. We compute local 3D rotation axes out of the 2D image silhouette of the rendered scene, using heuristics that combine the silhouette's image skeleton and depth in-

formation from the rendering's Z-buffer. Specifying such local rotation axes is simple and intuitive, requiring a single click and drag gesture, as the axes are automatically computed using the closest scene fragments rendered from the current viewpoint. Our method is simple to implement, using readily-available distance and feature transforms provided by modern 2D skeletonization algorithms; can handle 3D scenes consisting of arbitrarily complex polygon meshes (not necessarily watertight, connected, and/or of good quality) but also 3D point clouds; can be integrated in any 3D viewing system that allows access to the rendered Z-buffer; and works at interactive frame-rates even for scenes of hundreds of thousands of primitives. We demonstrate our method on several polygonal and point-cloud 3D scenes of varying complexity.

Several extension directions are possible as follows, apart from the user evaluation of the effectiveness of our skeleton-based rotation mechanism which will be discussed in the next Chapter. More cues can be used to infer more accurate 3D curve skeletons from image data, such as shading and depth gradients. Different simplification techniques for the inferred 3D skeletons can be proposed, thereby making the computed 3D rotation axes become more robust to noise present in the visualized shapes and, more importantly, able to capture the 'natural' axis of rotation implied by that shape at a given location. Finally, a challenging but interesting extension direction considers computing such 3D rotation axes directly from a volume-rendered visualization, by extracting salient structures in the visualization and reducing these to their skeletons.

4

EVALUATING THE EFFECTIVENESS OF THE SKELETON-AND-TRACKBALL INTERACTION TECHNIQUE

Abstract: In Chapter 3, we presented a method for interactive specification of rotations of 3D shapes and point clouds around axes determined by the underlying structure of such shapes. We showed there how our method is simple to implement, computationally efficient, and applicable to any 3D content that can be rendered in an OpenGL viewer. Its usage requires only simple point, click, and drag gestures. In this chapter, we study the effectiveness and ease of adoption of the method in practice. For this, we compare our method with classical trackball rotation, both in isolation and in combination, in a controlled user study. Our results show that, when combined with trackball, skeleton-based rotation reduces task completion times and increases user satisfaction, while not introducing additional costs, being thus an interesting addition to the palette of 3D manipulation tools [1].

4.1 INTRODUCTION

Three-dimensional content such as polygonal models or point clouds appear in a multitude of contexts and are targeted by a wide range of users. As such, having *efficient* and *effective* tools to explore such scenes is a major component that influences the success (or lack thereof) of applications that handle such content such as editors or content viewers.

In Chapter 3, we proposed a new such exploration mechanism targeted at specifying rotations around 3D axes defined by the content itself. In brief, the user points at a region of interest (part) of the viewed 3D shape, from which a local symmetry axis is computed. Next, one can rotate the shape around this axis with an interactively specified angle. This method allows an easy selection of parts and automatic computation of their approximate 3D symmetry axes, both done using the shape silhouette's 2D skeleton. The method handles any 3D scene, *e.g.*, polygon mesh or polygon soup, point-based or splat-based rendering, or combination thereof, without preprocessing; can be implemented using simple image processing operations; and works at interactive rates for scenes of hundreds of thousands of primitives. As such, we argue that our proposed method complies well with the desiderata of computational scalability, genericity, and simplicity of implementation, which

1 This chapter is based on the paper 'Skeleton-and-Trackball Interactive Rotation Specification for 3D Scenes' Zhai et al. (2022).

together cover the efficiency term mentioned in the beginning of this section.

However, the effectiveness of our method also needs to be studied. In Chapter 3, we mentioned that our skeleton-based rotation is not to be seen as a replacement, but a *complement*, of classical trackball rotation. This observation was made based on our own impressions gathered during the actual development and testing of our method in an application which availed of the traditional – and ubiquitous – virtual trackball mechanism. However, many concrete questions related to effectiveness are still open. We classify these into two groups, as follows.

To start with, how precisely the skeleton-based and trackball rotation mechanisms *relate* to each other is not yet known. Many possibilities exist, such as each mechanism behaving best for a particular type of manipulation and/or a particular type of 3D content. It is, however, also possible that one mechanism is consistently better than the other, regardless of the manipulation or content (shape) type. Knowing this is important to be able to next suggest a particular mechanism for a given use context in real applications.

Secondly, how the skeleton-based interaction mechanism is *received* by actual end users is another important question to be answered. Possible subquestions hereof are how users experience the mechanism in terms of ease of learning, ease of use, precision of the operations it supports, and overall satisfaction. Separately, it is useful to compare how users rate the skeleton-based interaction mechanism as compared to the trackball one. Answering such questions can point to focused directions for improving our proposal but also ways to deploy it for supporting specific operations in real applications.

In this chapter, we aim to answer several of the above questions by the following two contributions:

- We present the design and execution of a controlled user study aimed at gauging the added value of skeleton-based rotation when used against, but also combined with, trackball rotation;
- We analyze the results of our study to show that, when used together with trackball rotation, skeleton-based rotation brings in added value, therefore being a good complement, and not replacement, of trackball rotation.

The structure of this chapter is as follows. Section 4.2 presents a formative evaluation of our method. Section 4.3 presents an in-depth quantitative and qualitative user study that studies the hypotheses outlined by the formative study. Section 4.4 discusses the skeleton-based rotation and our findings regarding its best ways of use. Section 4.5 concludes the chapter.

4.2 FORMATIVE EVALUATION

To evaluate our method, we conducted first a *formative* evaluation. In this evaluation, only the authors of this work and a few other researchers, familiar with 3D interactive data visualization, were involved. This evaluation aimed at (a) verifying how the skeleton-based rotation practically works on a number of different 3D shapes; and (b) eliciting preliminary observations from the subjects to construct next a more in-depth evaluation study. We next present the results of this first evaluation phase. Section 4.3 details the second-phase evaluation designed using these findings.

The evaluation used the three shapes presented earlier in Chapter 3 in Figures 3.6 and 3.7. As described there (Sec. 3.4), these three shapes are quite different – a simple model of a hand consisting of a few parts, a more complex polygonal model of a ship having several tens of parts of different sizes and structures; and a point cloud. No explicit tasks were given in this evaluation apart from the general indication to (freely) explore and examine the shape from different viewpoints. Also, no explicit time limit was placed on the exploration.

We gathered several insights during our formative evaluation by free-form discussions with the participants – that is, without following a strict evaluation protocol based on tasks and quantitative responses. We summarize below the most important ones:

- Skeleton rotation works quite well for relatively *small* changes of viewpoint; more involved changes require decomposing the desired rotation into a set of small-size changes and careful selection of their respective rotation axes;
- Skeleton rotations seem to be most effective for *precise* rotations, in contrast to typical trackball usage, which works well for larger, but less precise, viewpoint changes;
- All participants stated that they *believe* that skeletons allow them to perform certain types of rotation easier than if they had used the trackball for the same tasks. However, they all mentioned that they do not feel that skeletons can *replace* a trackball. Rather, they believe a free combination of both to be most effective. Since they could only use the skeleton rotation (in our evaluation), they do not know whether (or when) this tool works better than a trackball;
- All participants agreed that *measuring* the added-value of skeleton rotation is very important for its adoption.

4.3 DETAILED EVALUATION – USER STUDY

The formative evaluation (Sec. 4.2) outlined that there is perceived added-value in the skeleton rotation tool, but this value needs to be actually measured before users would consider adopting the tool – either standalone or in combination with trackball. To deepen our understanding of how skeleton-based rotation works, and to answer the above questions, we designed and conducted a more extensive user evaluation. We next describe the design, execution, and analysis of the results of this evaluation.

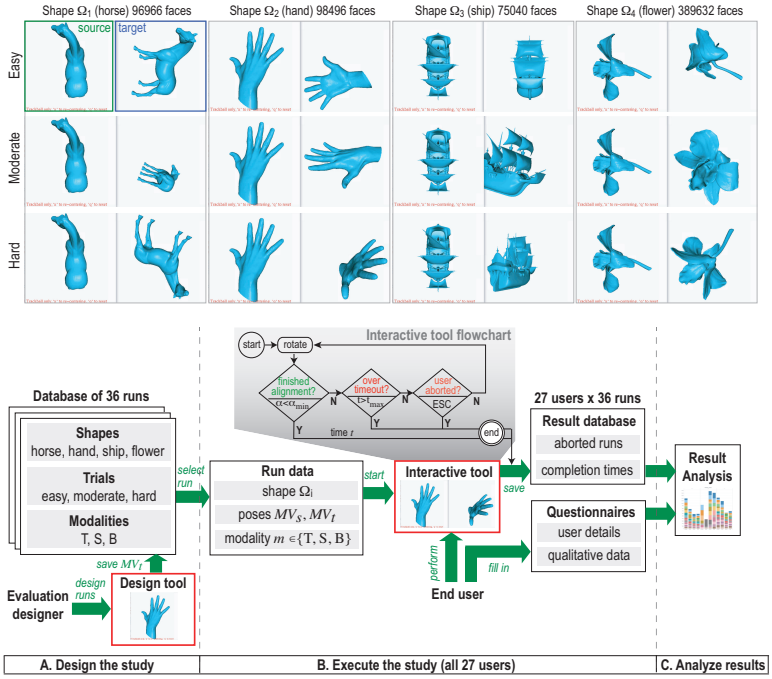


Figure 4.1: Top: User evaluation showing the 12 trials for one modality (Sec. 4.3.1). Each trial consists of a source window in which the user interacts to align the shape to match the target window. Bottom: Execution of end-to-end user evaluation. The use of our interactive tool in both design and evaluation modes is shown in red (Sec. 4.3.2).

4.3.1 *Evaluation design*

Tool: To assess how the skeleton rotation modality compares with the trackball modality, we designed an experiment supported by an interactive *tool*. The tool has two windows: The *target* window shows a 3D shape viewed from a viewpoint (pose) that is preselected by the evaluation designer. No interaction is allowed in this window. The *source*

window shows the same shape, which can be freely manipulated by the user via the skeleton (S), the trackball (T), or both tools (B), activated via the left, respectively right, mouse buttons. Both windows have the same resolution (512^2 pixels), use the same lighting and rendering parameters, and have a fixed position on the computer screen, to simplify usage during the experiment that invokes multiple runs of the tool. Besides rotation, the tool also allows panning and zooming. We also added an option to automatically zoom out to show the full extent of a shape. This eliminates the issues described in Sec. 3.3.3, *i.e.*, manipulations that move part of the shape outside the window. When in S mode, the tool shows the silhouette skeleton (black), nearest skeleton points (yellow), and estimated rotation axis (red) as explained earlier in Sec. 3.3.1 and shown *e.g.* in Fig. 3.3. The user can interactively tune the simplification level of the skeleton via the '+' and '-' keys, to show more or fewer branches from which to select a suitable rotation axis (*cf* Fig. 3.2).

Figure 4.1 (central inset) shows a flowchart of the tool's operation, which we detail next. Participants are asked to use the tool with each modality in turn (S, T, B) to align the source with the target – up to translation, which is deemed not important, since our goal is to investigate interactive tools for rotation specification. The tool continuously computes, after each motion of the mouse pointer, the value

$$\alpha = \arccos\left(\frac{Tr(MV_s \cdot MV_t^T) - 1}{2}\right), \tag{4.1}$$

where MV_s and MV_t are the 3×3 OpenGL rotation matrices (ignoring, thus, translation and scaling) corresponding to the pose of the shape in the source and the target, respectively; Tr is the matrix trace operator; and T denotes matrix transposition. The value $\alpha \in [0, 180]$ is the smallest rotation (around any axis) needed to obtain the target pose from the source pose (Belousov, 2016). Note that Eqn. 4.1 is sensitive to mirroring, which is desired, since rotations *cannot* cause mirroring. Alignment is considered *completed* when $\alpha < \alpha_{min}$; in practice, we set $\alpha_{min} = 15$ degrees. Also, note that Eqn. 4.1 only checks for *rotation*, and not scaling or panning, differences. This makes sense, since the tested modalities S, T, B control rotation only; scaling (zooming) and panning, though allowed to help users to inspect shapes, are not part of our evaluation, and perform identically with S, T, and B. During manipulation, the tool continuously displays the current value of α. This shows users how far away they are from the target rotation MV_t, thus, from completing a task. This feedback is useful when visual comparison of the source and target poses is hard to do.

Shapes: We use the alignment tool to evaluate the performance of the S, T, and B modalities on $N = 4$ *shapes* Ω_i, $1 \leq i \leq N$, shown in Fig. 4.1(top). Shapes were selected so as to be familiar, have a structure that exposes potential local-rotation axes, and have geometric com-

plexity ranging from simple (horse, hand) to complex (ship). The flower shape is of lower complexity than the ship; however, its manifold structure makes it particularly hard to understand and manipulate, since it looks quite similar from many viewpoints. All shapes use identical material properties and no opacity or textures, to favor uniform evaluation. We excluded the more complicated point-cloud shape (Fig. 3.7) used during formative evaluation (Sec. 4.2) since no more than five of our recruited subjects had the technical background needed to understand what such data means in the first place.

Task difficulties: For each shape, we use three target poses MV_t to capture three levels of difficulty of the alignment task:

- *Easy:* Alignment can be done by typically one or two manipulations, such as a rotation around one of the x or y window axes, or a rotation around a clearly-visible symmetry axis of the shape. For example, the blue-framed target in Fig. 4.1 can be obtained from the green-framed pose (left to it) by a single counterclockwise rotation of the horse with 90 degrees around the y axis or, alternatively, the rump's skeleton;
- *Hard:* Alignment requires multiple rotations around many different rotation axes; it is not easy to see, from the source and target, which would be these axes;
- *Intermediate:* Alignment difficulty is gauged as between the above two extremes.

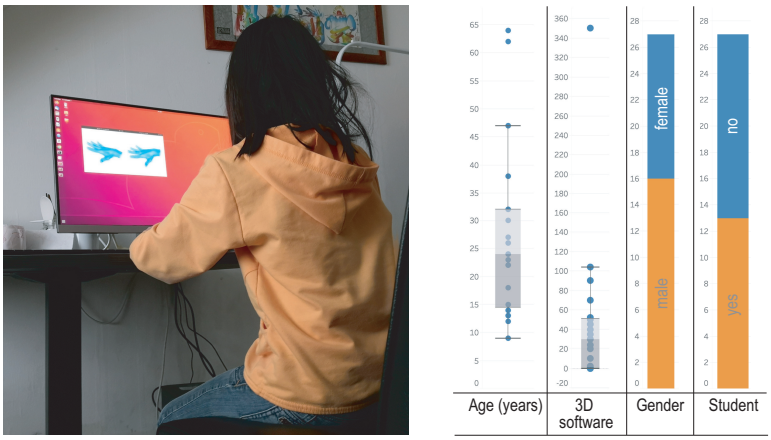
We call next the combination of shape Ω_i and start-and-end pose (MV_s, MV_t) a *trial*. Using multiple-difficulty trials aims to model tasks of different complexity. Trial difficulty was assessed by one of the authors (who also designed the actual poses MV_t) and agreed upon by the others by independent testing. We verified that users employing all three modalities could accomplish all trials within a time t lower than a predefined timeout $t_{max} = 120$ seconds.

Figure 4.1(top) shows the source (left window in each window-pair) and target (right window in the same pair) windows for the 12 trials spanning the 4 shapes using the T modality. Source windows show the currently-enabled modality in red text, to remind users how they can interact. We see, for instance, that the *easy* trial would require, in T mode, a simple 90-degree rotation around the y axis for the ship model, or around the main skeleton branch passing through the horse's rump for the horse model, respectively. In contrast, the *hard* task requires several incremental rotations for all modalities. The 12 trials use *identical* initial poses MV_s and target poses MV_t. That is, the user is asked to perform, for each shape, the same alignment $MV_s \rightarrow MV_t$ using all three modalities, thus ensuring that only the the target pose (endpoint

of manipulation) and, of course, the used modality, affect the measured execution time.

In total, we thus execute 12 trials ×3 modalities = 36 *runs*. For each run, we record the time needed for the user to complete it. If the user fails to perform the alignment within the allowed timeout, the run is considered *failed* and the user moves automatically to the next run. Users can at any time (a) abort a run by pressing 'ESC' to move to the next run; this helped impatient users who did not grasp how to perform a given alignment task and did not want to wait until the timeout; (b) abort the entire evaluation, if something goes entirely wrong; and (c) reset the viewpoint to the initial one (MV_s), to 'undo' all manipulations performed so far if these are deemed unproductive.

Pose design: The different target poses MV_t were designed in advance by us by using the S and T tools – intermixed – to freely change the shape's pose until obtaining the desired target poses, and stored, as explained, as 3×3 OpenGL rotation matrices.



a) b)

Figure 4.2: (a) Setup employed during the user evaluation. (b) Self-reported characteristics of the experiment participants. See Sec. 4.3.2.

4.3.2 *Evaluation execution*

Subjects: Twenty-seven persons took part in the evaluation. they self-report ages of 9 to 64 years (median: 24, average: 26.9); and gender being male (16) and female (11), see Fig. 4.2b. To gauge their experience with 3D manipulation, we asked them to report how many times a year they used 3D games and/or 3D design software. Both categories are reported in Fig. 4.2b as '3D software usage'. Results show a median of 30 times, with the minimum being zero (never) and the maximum being

basically every day. From these data we conclude that most participants should have a good practical mastery of 3D manipulation. From the 27 participants, 13 were students in fields as diverse as Computer Science, social science, medicine, economy and society, and mathematics; the other 14 were primary or secondary school pupils (6) or employed in various liberal professions (8). All participants reported no color blindness issues. All except one were right-handed. They all reside in the Netherlands or Belgium. Communication during the training and experiment was done in the native language of each participant by a (near-)native speaker. For participants with limited English proficiency, all English material (tutorial, questionnaires) was transcribed by the trainer.

Workflow: Participants followed the evaluation workflow showed in Fig. 4.1(bottom). First, we created the information needed to execute the 36 runs (Fig4.1(bottom, A)), as explained in Sec. 4.3.1. Next, participants were given access, prior to the actual experiment, to a web tutorial which describes both S and T tools in general, and also allows users to practice with these tools by running the actual application to execute some simple alignment tasks. No statistics were collected from this intake phase. After intake, users were asked if they felt interested in, and able to follow, the tutorial. This intake acted as a simple filter to separate users with interest in the evaluation (and potential ability to do it) from the rest, so as to minimize subsequent effort. Seven persons dropped from the process due to lack of general computer skills (1 user), one too young (6 years), one too old (82 years), and four due to technical problems related to remote-deployment of the tool. These persons are not included in any of the statistics further on, nor in Fig. 4.2b. For clarity, we did not ourselves drop the too-young and too-old users from the study – the decision to quit was their own based on their own assessment of not being able to complete the tasks being asked.

Next, a trainer (role filled by different co-authors) took part in a *controlled session* where they explained to either individual participants or, when social distancing rules due to the Corona pandemic were not applicable, to groups of participants how the tool works and also illustrated it live. The aim of this phase was to refine the knowledge disseminated by the web tutorial and confirm that participants understood well the evaluation process and tooling. Participants and trainers used Linux-based PCs (16 to 32GB RAM) with recent NVidia cards, wide screens, and a classical two-button mouse. To maximize focus on the experiment, no application was run on screen during the evaluation besides the two-window tool described in Sec. 4.3.1. Training took both the in-person form (with trainer and user(s) physically together), and via TeamViewer or Skype screen sharing, when social distancing rules mandated separation. Training took between 20 and 40 minutes per user, and was done until users told that they were confident to use the tool to manipulate

both a simple model and a complex one via all three modalities (S, T, B). During this phase, we also verified that the tool runs at real-time frame rates on the users' computers so as to eliminate confusing effects due to interaction lag; and that the users did not experience any difficulty in using the keyboard shortcuts outlined in Sec. 4.3.1.

After training, and confirmation by participants that they understand the evaluation tool and tasks to be done, participants started executing the 36 runs (Fig. 4.1(bottom, B). They could pause between runs as desired but not change the orders of the runs. Figure 4.2a shows the setup used during the evaluation by one of the actual participants; notice the two-window interaction tool on the screen. At the end, the results of all 36 runs – that is, either completion time or run failure (either by timeout or user abortion) – were saved in a database with no mention of the user identity. Next, users completed a questionnaire covering both personal and self-assessment data and answers to questions concerning the usability of the tool. Both types of results (timing data and questionnaires) were further analyzed (Fig. 4.1(bottom, C)), as described in Sec. 4.3.3.

4.3.3 *Analysis of results*

We next present both a quantitative analysis of the timing results and an analysis of the qualitative data collected via questionnaires.

4.3.3.1 *Analysis of timing results*

A most relevant question is: How did performance (measured in completion time and/or number of aborted runs) depend on the interaction *modality* and *shape*? Figure 4.3a shows the average completion time, for the *successful* runs, aggregated (over all users) per modality and next per shape. User identities are categorically color-coded for ease of reading the figure. Median and interquartile ranges for each modality are shown by black lines, respectively gray bands. We see that the S modality is significantly slower than T. However, the B modality is faster than T, both as median and interquartile range, and also for each specific shape. This is an interesting observation, as it suggests that, in B mode, users did gain time by using S only for some specific manipulations for which T was hard to use. A likely explanation for this is that the B modality was always used last during the trials. Hence, when in B mode, users could discover the situations when S outperformed T, and switch to S in those cases to gain time. We will analyze this hypothesis further below.

Figure 4.3b shows the number of *failed* runs per modality, shape, and user. These are largest for the S modality. This tells again that S cannot be used *alone* as a general-purpose manipulation tool. If we combine this insight with the total times per shape (Fig. 4.3a), we see that the perceived difficulty of the task varies significantly over both shapes and modalities: T and S behave quite similarly, with *horse* and *ship* being eas-

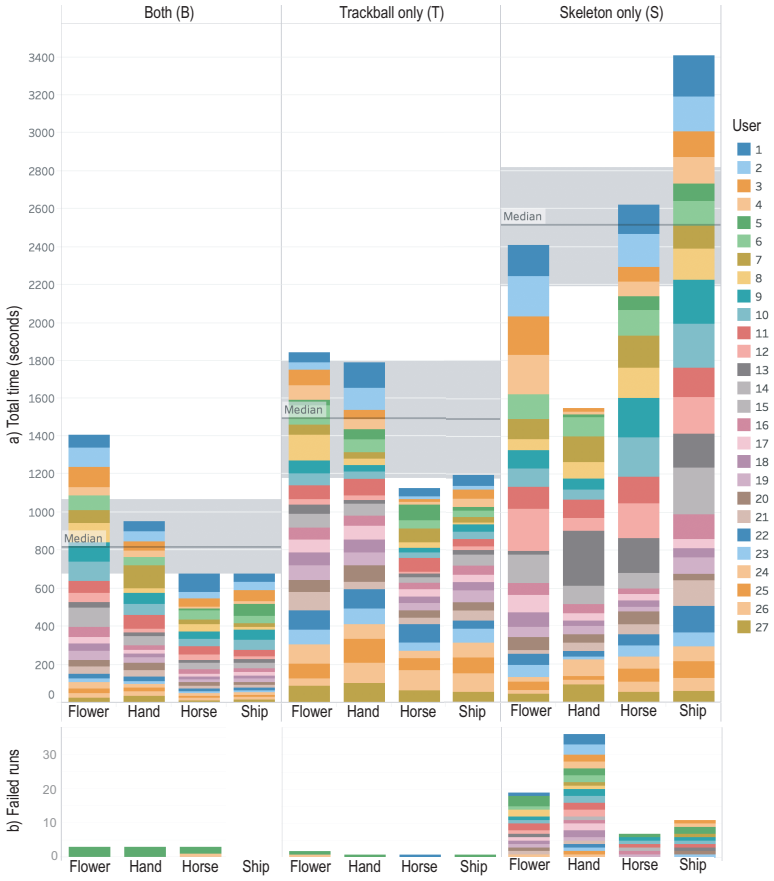


Figure 4.3: Completion time (a) and number of failed runs (b) per modality and shape, all users. See Sec. 4.3.2.

ier to handle and *flower* being the hardest. In contrast, *hand* seems to be the hardest to handle by the S modality, as it has most aborted runs. Upon a closer analysis, we found that the pose used by the 'hard' trial for *hand* (see the respective image in Fig. 4.1(top)) is quite easy to achieve with T (and thus also B), but quite difficult to obtain using S, since it implies, at several points, performing a rotation around an axis orthogonal to the hand's palm, for which no skeleton line exists in the silhouette. The second-hardest shape for S is *ship*. Analyzing the users' detailed feedback showed us that *ship*'s complex geometry produces a wealth of potential rotation axes with quite different angles, which makes the users' choice (of the optimal rotation axis) hard. This happens far less for the other simpler-structure shapes. Separately, Fig. 4.3b shows that the number of aborted runs in B mode is far lower than that in S mode, being practically the same as for T mode. This, and the fact that B mode is fastest, reinforces our hypothesis that users employ the S tool in B mode only for very *specific* manipulations and revert to T for all other operations. Hence, S works best as a complement, not a replacement, of T.

Figure 4.4a introduces additional information in the analysis by showing how the average times vary over the three task *difficulty* levels (easy, moderate, hard, see Sec. 4.3.1). For all shapes and modalities, the task labeled 'easy' by us is, indeed, completed the fastest. The other two difficulty levels are, however, not significantly different in execution times. We also see that effort (time) is distributed relatively uniformly over all difficulty levels for all shapes and modalities. This indicates that there is no 'outlier' task or shape in our experiment that would strongly bias our evaluation's insights.

Finally, we examine the data from a *user-centric* perspective. Figure 4.4b shows the total time per user, split per modality, with the fastest users at the right and the slowest at the left. We see a quite large spread in performance, the fastest user being roughly 2.5 times faster than the slowest one. We see that the T modality does not explain the big speed difference – the red bars' sizes do not correlate with the total time. In contrast, the blue bars show an increase when scanning the chart right-to-left, at the 8^{th} leftmost bar – meaning that the 8 slowest users needed clearly more time to use the B modality as opposed to the remaining 19 users. Scanning the graph right-to-left along its orange bars shows a *strongly* increasing bar-size. That is, the main factor differentiating slow from fast users is their skill in using the *S tool*. We hypothesized that this skill has to do with the users' familiarity with 3D manipulation tools. To examine this, we show a scatterplot of the average time per user (all trials, all shapes) *vs* the user's self-reported number of days per year that one uses 3D computer games or 3D creation software (Fig. 4.4c). All points in the plot reside in the lower range of the y axis, *i.e.*, all users report under 100 days/year of 3D tool usage, except user 12 who indicated 3D gaming daily. The computed correlation line shown in the

figure ($R^2 = 0.0022$, $p = 0.813$) indicates a negligible inverse correlation of average time with 3D software usage. Hence, our hypothesis is not confirmed. The question what determines the variability in users' average completion times is still open.

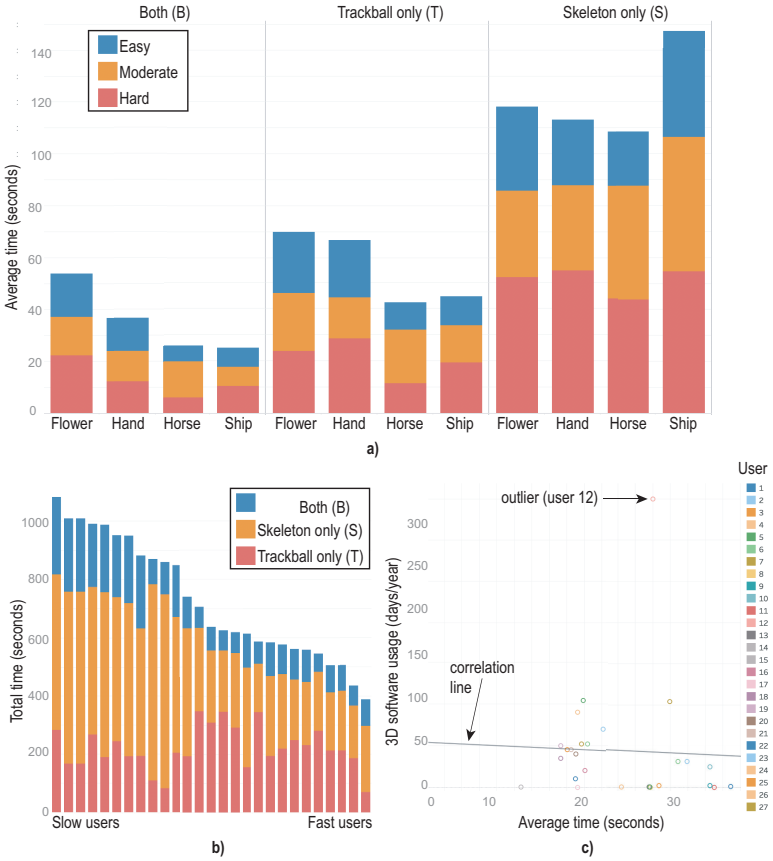


Figure 4.4: a) Average completion time per difficulty levels, modality, and shape. b) Total time for all users, from slowest to fastest, split per modality. c) Correlation of average time (all runs) with users' frequency of 3D software usage. See Sec. 4.3.2.

4.3.3.2 *Questionnaire results*

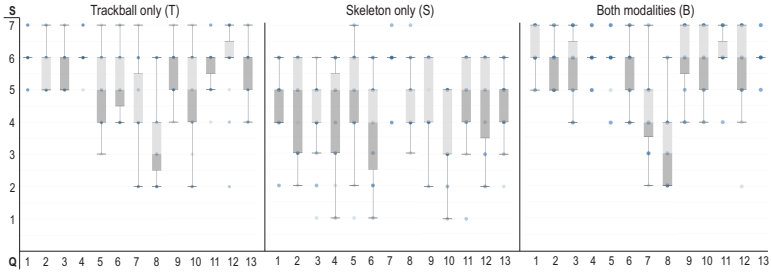
As mentioned at the beginning of Sec. 4.3.2, users completed a questionnaire following the experiment. They were asked to answer 13 questions concerning their experience with each of the three modalities (T, S, B) using a 7-point Likert scale S (1=strongly disagree, 2=disagree, 3=disagree somewhat, 4= no opinion, 5=agree somewhat, 6=agree, 7=strongly agree). An extra question (Q14) asked which of the three

modalities users prefer *overall*. Figure 4.5(bottom) shows these 14 questions. Here, 'tool' refers to the modality being evaluated. Following earlier studies that highlight that user *satisfaction* is not the same as user *efficiency* or *effectiveness* when using interactive tools (Frokjaer et al., 2000; Partala, 1999), we included questions that aim to cover all these aspects. Users could also input free text to comment on their perceived advantages and limitations of all three modalities or any other remarks.

Figure 4.5(top) shows the aggregated answers for Q1..Q13 for each of the three modalities with box-and-whisker plots (box shows the interquartile range; whiskers show data within 1.5 times this range). We see that the S modality ranks, overall, worse than the T modality, except for accuracy (Q5). Accuracy (Q5) can be explained by the fact that users need to control a *single* degree of freedom with S – the rotation angle – but two degrees of freedom with T. In other words, once a suitable rotation axes is chosen, S allows one to precisely specify the rotation angle around this axis. We also see that S helps completing the task less often than T (Q10), which matches the failure rates shown in Fig. 4.3b. However, the B modality ranks in nearly all aspects better than both T and S. This supports our hypothesis that S best *complements*, rather than replaces, T. An interesting finding are the scores for Q8 and Q6, which show that B was perceived as less tiring to use, and needing fewer steps to accomplish the task respectively, than both T and S. This matches the results in Fig. 4.3a that show that B is faster than both T and S – thus, arguably less tiring to use. For Q14, 22 of the 27 users stated that they prefer B overall, while the remaining 5 users preferred T, with none mentioning S as the highest-preference tool. As for the previous findings, this strongly supports our hypothesis that the S and T modalities work best when combined.

From the free text that captures the user's comments on the perceived advantages and limitations of all three modalities, we could distil several salient points. For space constraints, we list only a few below:

- **Trackball (T):** Several users praised T for being "easy to use". However, users also complained about trackball being imprecise for performing fine adjustments;
- **Skeleton (S):** This modality was mentioned as better than the other two by only a few users, and specifically for the horse, hand, and flower models, because of their clear and simple skeletons, which allow one to intuitively rotate the shape around its parts ("easy to turn the hand around a finger"'; "easy to turn the horse around a leg"; "S helps to turn the flower around its stem"). However, several users mentioned advantages of S when used in combination with T. These are discussed below;
- **Both (B):** Overall, this modality received the most positive comments. It was deemed the "most accurate"; and "feeling quick to



Q1	The tool met my needs for performing the alignment task
Q2	The tool worked as expected (after following the training)
Q3	The tool helped me be more effective than the two other tools
Q4	The tool was easy to use
Q5	The tool was accurate
Q6	The tool requires the fewest steps (compared to the other two) to accomplish my goals
Q7	I felt that I have to think carefully to get a good result with this tool
Q8	The tool was tiring to use
Q9	Both occasional and regular users would like the tool
Q10	I can use the tool successfully every time
Q11	I learned to use the tool quickly
Q12	I easily remember how to use the tool
Q13	I am satisfied with the tool
Q14	Which tool (T, S, or B) do you overall prefer?

Figure 4.5: Results of 13-point user questionnaire for the three modalities. Questions are shown below the charts. See Sec. 4.3.3.2.

> use when we have two methods [to choose from]". Specifically, users noted that B is "good for doing final adjustments / fine tuning the alignment" and that "S helps T to getting the desired result easily" and "I started with T and used S for final touches". One user also commented: "I work as a graphic designer with a lot of 3D tools; I see how S helps me by providing a lot of control when rotating, and I would love to have this tool along my other manipulation tools in my software [...] but I would not use it standalone".

Summarizing the above, we see that our initial hypothesis that the S modality helps (complements) T for precision tasks is largely supported by user experience.

4.4 DISCUSSION

The evaluation described in Sec. 4.3 confirmed the insights elicited from the earlier formative study (Sec. 4.2), *i.e.* that skeleton rotation is best for precise, small-scale, final alignment touches; and that skeleton rotation best works as a *complement*, and not replacement, of trackball rotation. The latter point was supported by all types of data from our evaluation – task timing, scores assigned by users to evaluation questions, and free-form text feedback. The same data shows that users rank the combined modality (B) as better than both S and T modalities taken separately. The user scores also show that, overall, the combined modality is easy to learn and use (Fig. 4.5, Q2-4-8-11-12). Put together, all above support our claim of added value for the skeleton-based rotation technique.

Besides the above results, the user study also unveiled several questions which we cannot fully answer:

User performance: There is a large variability of user performance, measured as task success rates and completion times (see Fig. 4.4b and related text). We cannot explain this variability by differences in the experiment setup, previous user familiarity with 3D manipulation, amount of training with the evaluated tool, or other measured factors. This variability may be due to user characteristics which the self-reported variables (Fig. 4.2b and related text) do not capture; to the high heterogeneity of the user population; but also due to dependent variables which we did not measure, *e.g.*, how often did users use the skeleton simplification level (Sec. 4.3.1) to produce suitable skeletons for generating rotation axes. Repeating the experiment with a more homogeneous population and more measured variables would help answering this question.

Applicability: An important limitation of our study is that, for the B modality, we did not measure how (much) the task was completed

using each *separate* modality, *i.e.*, S and T. The formative study (Sec. 4.2), textual user feedback for the controlled experiment, and our observation of the users during the experiment jointly show that, in most cases involving moderate or hard tasks, trackball was *first* used to obtain a viewpoint roughly close to the target one, which was next fine-tuned using skeleton. This is fully in line with our initial design ideas (see Fig. 3.1 and related text) and also with earlier findings on what trackball best works for Jacob and Oliver (1995); Partala (1999); Henriksen et al. (2004a). However, understanding more precisely which are the rotation types that skeleton best supports would greatly help to improve the combined modality by *e.g.* suggesting this modality to the user when it appears fit, and/or conversely, blocking this modality when it does not match the task at hand.

Study limitations: A limitation that, by now, should have become apparent to the reader is that the compared rotation modalities – virtual trackball and skeleton-based – specify two different *kinds* of rotation. Indeed, virtual trackball specifies a rotation around a *point*; in contrast, the skeleton modality specifies a rotation around an *axis*. As such, the differences in usage of the two modalities can be partially explained by the intention of the user: One would prefer the trackball when wanting to rotate a shape around a given point, and respectively the skeleton modality when wanting to rotate a shape around a given axis. One could argue that a fairer comparison would be that between our proposed skeleton modality and other modalities for specifying rotation around 3D axes. We have considered this possibility. However, we have observed that mechanisms to specify rotations around 3D axes are, in general, quite complex to implement in terms of user interface – see also the discussion in Sec. 3.2 – unless one would use multitouch screens. As such, an important driver in our evaluation was to select a rotation mechanism that is comparably easy to use to our skeleton modality, and also have both modalities able to run on classical set-ups involving simple mouse-and-keyboard input. The virtual trackball emerged as a clear candidate concerning these requirements, which is why we chose it for executing the user evaluation. An evident direction of improving this comparison is to have our skeleton modality implemented on multitouch devices and compared via an user evaluation with easy-to-use rotation specification mechanisms for such devices such as FI3D and its extensions (Yu et al., 2010; Guo et al., 2017; Yu and Isenberg, 2009).

Besides the above-mentioned aspects, our study (Sec. 4.3) has further limitations: It uses only four shapes that cannot capture the rich distributions of 3D shapes that need manipulation. Also, it only covers the task of rotating from an initial pose to a given final pose. Yet, manipulation is also used for free exploration and/or design actions which do not require reaching a predefined pose. It is unclear how to *quantitatively* measure the added value of interaction tools in such contexts,

beyond qualitative user-satisfaction questionnaires (Henriksen et al., 2004a). Also, we cannot exclude learning effects between the trials that address the same task with different modalities. Finally, what is the exact added-value of all the rotation-specification improvements (Sec. 3.3.3) was not currently measured. Exploring all these directions is left to future work.

4.5 CONCLUSION

In this Chapter, we described the execution of a user evaluation study aimed to compare and contrast the 3D rotation specification mechanism using image skeletons introduced in the previous Chapter with other established mechanisms for specifying 3D rotations. We measured the added value of our proposed rotation technique by a formative study (to elicit main concerns from users) followed by a controlled user study. Results showed that, when combined with the classical virtual trackball rotation, our method leads to better results (in terms of task completion times) and higher user satisfaction than trackball rotation alone. Also, our method is easy to learn and does not carry a significant learning or execution cost for the users, thereby not increasing the costs of using standard trackball rotation.

Several future work directions are possible. More cues can be used to infer more accurate 3D curve skeletons from image data, such as shading and depth gradients, leading to more precise rotation axes. Such data-driven cues could be also used to better control the rotation, and also suggest to the user which of the two modalities (skeleton-based or trackball rotation) are best for a given context. Separately, we aim to deploy our joint skeleton-and-trackball rotation tool on touch displays (single or multiple input) and evaluate its effectiveness in supporting domain experts to perform 3D exploration for specific applications, such as the astronomical data exploration outlined in Sec. 4.2.

5

USING MULTIPLE ATTRIBUTE-BASED EXPLANATIONS OF MULTIDIMENSIONAL PROJECTIONS TO EXPLORE HIGH-DIMENSIONAL DATA

Abstract: Multidimensional projections (MPs) are effective methods for visualizing high-dimensional datasets to find structures in the data like groups of similar points and outliers. The insights obtained from MPs can be amplified by complementing these techniques by several so-called explanatory mechanisms. We present and discuss a set of six such mechanisms that explain MPs in terms of similar dimensions, local dimensionality, and dimension correlations. We implement our explanatory tools using an image-based approach, which is efficient to compute, scales well visually for large and dense MP scatterplots, and can handle any projection technique. We demonstrate how the provided explanatory views can be combined to augment each other's value and thereby lead to refined insights in the data for several high-dimensional datasets, and how these insights correlate with known facts about the data under study[1].

5.1 INTRODUCTION

Multidimensional Projections (MPs) are among the methods of choice for visualizing high-dimensional data, as they scale well in terms of the number of data points and data dimensions that they can show on a given screen space. They are useful in exploring the data structure, specifically in identifying similar sets of points and outlier points. However, understanding what, in terms of data values, ranges, or relations between dimensions, makes these structures appear in the projection (and thus, in the data) is not trivial. Several mechanisms exist to this end, as follows. *Global* explanations, such as biplot axes (Greenacre, 2010; Gower et al., 2011) and axis legends (Broeksema et al., 2013; Coimbra et al., 2016) show how dimensions influence an entire projection, and as such cannot, in general, explain the formation of local patterns like clusters. Linked views and tooltips show *local* explanations, but require one to manually select structures of interest in the projection (Pagliosa et al., 2015; Joia et al., 2011; Rauber et al., 2015). *Image-based* techniques (Aupetit, 2007; Schreck et al., 2010; Martins et al., 2014) display

1 This chapter is based on the papers 'Enhanced Attribute-Based Explanations of Multidimensional Projections' (van Driel et al., 2020) and 'Using Multiple Attribute-Based Explanations of Multidimensional Projections to Explore High-Dimensional Data' (Tian et al., 2021c).

local explanations everywhere on the projection, not requiring one to select specific point subsets. They scale well visually and computationally, are clutter-free, and can generically handle any high-dimensional dataset.

da Silva et al. (2015) proposed an image-based explanation that colors every projection point by the dimension that contributes most to the similarity of data points in that neighborhood or, more technically put, the dimension which has the lowest variance over the respective neighborhood, or the dimension which contributes the least to the Euclidean distance between the points across that neighborhood. However, as noted in this work, having a *single* explanation is usually not sufficient to understand what points that are nearby in a projection, *e.g.*, which form a visual cluster, share among themselves. This is not surprising since there are many dimensions in the input dataset that the MP technique is jointly analyzing to create the respective projection.

To alleviate this situation, we create additional explanatory views which capture other aspects of the high-dimensional data at hand, as follows. First, we use principal component analysis (PCA) to analyze point neighborhoods to deduce and encode the local (intrinsic) dimensionality of the data. This allows users to separate regions of high intrinsic dimensionality in the projection (hard to explain by just a few dimensions) from low-dimensionality regions where such explanations are feasible. Secondly, we analyze point neighborhoods to detect and depict strong linear relationships (not captured by PCA) between dimensions. Our techniques complement existing mechanisms for projection explanation such as the attribute variance of Da Silva *et al.*, can be computed efficiently on the GPU, and can be applied generically on any high-dimensional dataset visualized by any MP technique.

Our work offers six explanatory views (distance contribution, variance, three ways to view dimensionality, and correlation) to explore MPs, arguing that more explanations would provide more insights in the data. However, how these five views can be combined, in practice, to explore real-world data, and how the obtained findings match ground-truth information about such data, are additional important questions to be answered to support our proposal. To answer such questions, we show several examples of how our five proposed views can be combined in a visual analytics fashion to find relevant insights in five real-world high-dimensional datasets that cannot be found using a single view. We also correlate the obtained insights with ground-truth information independently extracted by other researchers from three of these datasets.

Our six-view toolset depends on multiple technical settings such as hyperparameters controlling the computation of the various explanations involved in the views, *e.g.*, local neighborhood size, but also, most evidently, the choice of the MP technique used to generate the projections in the first place. To shed more light on how our produced explanations are to be interpreted, we study and discuss how they depend on

the values of these hyperparameters across a number of well-known MP techniques.

The structure of this chapter is as follows. Section 5.2 presents related work. Section 5.3 details our six explanatory views (including the original two proposed by da Silva et al. (2015)). Section 5.4 shows how the total set of six views can shed insights on projections of non-synthetic datasets, which we next correlate with available ground-truth information. Section 5.5 discusses our techniques. Section 5.6 concludes the chapter.

5.2 RELATED WORK

We start introducing a few notations. Let $D = \{\mathbf{x}_i\} \subset \mathbb{R}^n, 1 \leq i \leq N$, be a n-dimensional dataset with points $\mathbf{x}_i = (x_i^1, \ldots, x_i^n)$, also called samples or observations. We call the vectors $\mathbf{X}_j = (x_1^j, \ldots, x_N^j)^T, 1 \leq j \leq n$, the dimensions of D, also known as variables or attributes. Hence, D can be seen as a matrix of N rows (samples) and n columns (dimensions). A *projection* is a function $P : D \rightarrow \mathbb{R}^m$, $m \ll n$, which maps a high-dimensional point $\mathbf{x}$ to a low-dimensional one $P(\mathbf{x})$. In practice, $m \in \{2, 3\}$, so projecting an entire dataset D, denoted by $P(D) = \{P(\mathbf{x}) | \mathbf{x} \in D\}$, yields a 2D or 3D scatterplot. Projections aim to place points that are similar in D close to each other in $P(D)$ to enable users to recover the structure of D from the scatterplot $P(D)$. Similarity can be computed based on $\mathbb{R}^n$ distances (Tenenbaum et al., 2000; De Silva and Tenenbaum, 2004; Joia et al., 2011) or $\mathbb{R}^n$ neighborhoods (van der Maaten and Hinton, 2008; McInnes et al., 2018). Recent surveys provide more details on the technicalities of MPs (Nonato and Aupetit, 2018; Espadoto et al., 2019). In our work next, P can be any projection technique chosen by the user as desired or demanded by one's application context.

Explanatory techniques for projections aim to enrich the bare scatterplot $P(D)$ with additional information that guides the user in interpreting $P(D)$. We classify such techniques as observation-centric, dimension-centric, and hybrid, as follows.

5.2.1 *Observation-centric explanations*

These techniques aim to provide information about specific projection *observations* $P(\mathbf{x})$. Many such techniques aim to show the errors produced by the projection function P measured by *e.g.* normalized stress (Joia et al., 2011; Martins et al., 2014), correlation (Geng et al., 2005), Shepard diagrams (Joia et al., 2011), trustworthiness (Venna and Kaski, 2006b), continuity (Venna and Kaski, 2006b), neighborhood hit (Paulovich et al., 2008), distance consistency (Sips et al., 2009), ranking discrepancy (Lee and Verleysen, 2009; Lueks et al., 2013), projec-

tion precision score (Schreck et al., 2010), stretching and compression (Aupetit, 2007; Lespinats and Aupetit, 2011), and class consistency metrics (Tatu et al., 2010). Continuity and trustworthiness are closely related to the so-called missing neighbors, respectively false neighbors, of a projected point $P(\mathbf{x})$ (Martins et al., 2014). For a recent survey that discusses most above metrics, we refer to Nonato and Aupetit (2018).

Error metrics can be computed at three aggregation levels. *Global* errors generate a single (scalar) value for an entire scatterplot $P(D)$, so they help gauging the quality of such a scatterplot, but do little in explaining it. *Point-pair* errors quantify the projection error of a point pair $(P(\mathbf{x}), P(\mathbf{y})) \in P(D) \times P(D)$ and can be rendered as Shepard diagrams (Joia et al., 2011) or line plots simplified by edge bundling (Martins et al., 2014). *Point-neighborhood* errors quantify the projection error of a point $P(\mathbf{x}) \in P(D)$ with respect to all its neighbors in $P(D)$ or, alternatively, all neighbors of $\mathbf{x} \in D$. These are further visualized using heatmaps (Schreck et al., 2010; Martins et al., 2014) or Voronoi diagrams (Aupetit, 2007; Lespinats and Aupetit, 2011), thereby informing the user about projection problems at the location of every scatterplot point. This further assists one in determining where, and how much, one can trust a projection. However, such techniques cannot explain *why* certain points are projected close to each other (or not).

5.2.2 *Dimension-centric explanations*

These techniques show how the *dimensions* $\mathbf{X}_j$ of a dataset D relate to the scatterplot. The simplest, and still most used, dimension-centric explanation colors a scatterplot by the values of a selected dimension $\mathbf{X}_j$. This explains specific groups of points in the scatterplot by that dimension's values. Several dimensions can be used via interaction or small multiples. Yet, this approach cannot easily handle more than a few dimensions, leaving their selection to the user. Biplot axes (Greenacre, 2010; Gower et al., 2011) involve all dimensions in the explanation by drawing n lines atop of the scatterplot $P(D)$, each indicating the embedding of one of the dimensions $\mathbf{X}_j$ in the projection space $\mathbb{R}^m$. Axis legends (Oeltze et al., 2007; Broeksema et al., 2013) take a different route, by explaining how the n dimensions map to the 2D scatterplot's x and y axes using bar charts. Both biplots and axis legends have been generalized to explain also 3D projections and nonlinear projections (Coimbra et al., 2016).

All above dimension-centric explanations act as generalizations of the classical axis labels present in 2D Cartesian scatterplots – that is, they allow users to see which are the values of one or multiple dimensions that determine the *overall* projection shape. However, they do not *explicitly* connect the explanations to individual scatterplot points or point groups, leaving this to be done (visually) by the user. In contrast, observation-centric techniques explicitly mark individual points by the

provided explanations (*e.g.*, errors); however, such techniques do not involve dimensions in the explanation.

5.2.3 *Hybrid explanations*

Hybrid techniques aim to join the strengths of observation-centric and dimension-centric ones. The simplest form involves brushing points to show their attributes in a tooltip. More involved techniques involve interactively selecting and/or modifying specific *points* S in the projection. By next arranging $P(D) \setminus S$ around S, one can explain $P(D) \setminus S$ in terms of (known) attribute values of S. The VIBE system (Olsen et al., 1993) allows selecting and placing points of interest (POIs) in the 2D projection space according to one's mental map of how the respective data samples relate to each other. The remaining data points are projected based on their similarity to POIs. A similar approach is proposed in Joia et al. (2011) and by the ForceSPIRE text visualization system (Endert et al., 2012). The "dust and magnets" technique (Yi et al., 2005) extends these interaction metaphors by allowing users to interact with both POIs and data points, using animation to map the data-to-POI similarities. Interaction also supports navigating through a space of 2D scatterplots (whose axes are directly explained by their dimensions) created from the high-dimensional data (Piringer et al., 2004; Elmqvist et al., 2008). Pagliosa *et al.* propose a 'projection inspector' that offers several such interactive exploratory mechanisms. Interactive techniques are very powerful in providing 'details on demand' (on both observations and dimensions) to the user. However, they require interaction effort, and also cannot explain an *entire* projection, but rather the point(s) interacted with.

Image-based techniques, also known as dense maps, are a different hybrid approach. These rasterize the 2D projection space $\mathbb{R}^2$ and synthesize, for each pixel $\mathbf{p}$, an explanation based on the points in $P(D)$ nearest to $\mathbf{p}$. This space-filling approach allows a large amount of information to be conveyed; and removes issues of observation-centric techniques caused by overlapping points in $P(D)$. Da Silva *et al.* create dense maps where pixel hues encode the dimension that best explains the similarity of points in $P(D)$ close to each pixel, and brightness encodes the explanation confidence (da Silva et al., 2015). We extend this technique with explanations of the local dimensionality of data and dimension correlations. We detail both above techniques in Sec. 5.3.

Dense maps have been used to explain projection errors (Martins et al., 2014; Schreck et al., 2010; Lespinats and Aupetit, 2011)). Rodrigues *et al.* used dense maps to visualize the decision zones of classifiers of high-dimensional data (Rodrigues et al., 2019). Like us and Da Silva *et al.*, they also use pixel hues and luminances to encode a classifier's decision, respectively decision confidence, at a data point $\mathbf{x}$ mapping to a pixel $P(\mathbf{x})$. Our goals are different, as we aim to explain a dataset in terms of its *dimensions*, rather than a classifier in terms of its *decisions*.

5.3 EXPLANATORY MECHANISMS

The image-based explanatory techniques introduced in Sec. 5.2.3 exploit the distance or neighborhood preservation property of MPs: Let $\nu_i \subset P(D)$, $\nu_i = \{\mathbf{y} \in P(D) | \ \|\mathbf{y} - \mathbf{y}_i\| \leq \rho\}$, be a neighborhood of size ρ of scatterplot points $\mathbf{y}$ centered at $\mathbf{y}_i$. Since points in ν_i are, by construction, close, and since P is expected to (reasonably) preserve similarities, the points $\mu_i \in D$ that project to ν_i are expected to be similar. Hence, it makes sense to compute an *explanation* of μ_i and next visually encode this on all scatterplot points $\mathbf{y}_i$.

Da Silva *et al.* propose two such explanations (da Silva et al., 2015). Let $\lambda^j_{\mathbf{x},\mathbf{x}'} = \|\mathbf{x}^j - \mathbf{x}'^j\|^2_1 / \|\mathbf{x} - \mathbf{x}'\|^2_n$ be the contribution of dimension j to the distance between two points $\mathbf{x}$ and $\mathbf{x}'$ in D, where $\|\cdot\|_k$ is the Euclidean distance in $\mathbb{R}^k$. This point-pair contribution is extended to neighborhoods μ_i by averaging the local contributions of $\mathbf{x}_i$ and all its neighbors, as

$$\overline{\lambda}^j_i = \sum_{\mathbf{x} \in \mu_i} \lambda^j_{\mathbf{x},\mathbf{x}_i} / |\mu_i|, \tag{5.1}$$

where $|\cdot|$ denotes set size. These average contributions are next normalized as

$$\lambda^j_i = \frac{\overline{\lambda}^j_i / \gamma^j}{\sum_{j=1}^n \left(\overline{\lambda}^j_i / \gamma^j\right)}, \tag{5.2}$$

where the normalization γ^j is the contribution $\overline{\lambda}^j$ of dimension j of the full dataset D (see Eqn. 5.1) with respect to its centroid. Since normalized, $\lambda^j_i \in [0, 1]$, with lower values indicating dimensions that contribute *little* to distances in μ_i, *i.e.*, explain well why points in μ_i are *similar*. Da Silva *et al.* also propose an alternative to Eqn. 5.2 by computing the relative variance ω^j_i of dimension j over the neighborhood μ_i as

$$\omega^j_i = \frac{LV^j_i / GV^j}{\sum_{j=1}^n (LV^j_i / GV^j)}, \tag{5.3}$$

where LV^j_i is the variance of dimension j for all points in μ_i, normalized by the variance GV^j of the same dimension j over all points in D. Just as λ^j_i, $\omega^j_i \in [0, 1]$, with lower values telling dimensions that vary little in a neighborhood.

The scatterplot $P(D)$ is explained by color-coding its points by the C dimensions that have overall low values of λ^j_i (or ω^j_i, depending on the user's choice) over all points. C is set to a low value, *e.g.* 8, since categorical colormaps should be small. Luminance is used to encode the *confidence* in the visual explanation: If j is the dimension picked to

color point i, confidence κ is computed as the sum of λ_i^j (or ω_i^j) values for all points in the neighborhood μ_i, normalized by the sum of the same terms over *all* dimensions over μ_i. If neighbors of point i are best explained by the same dimension j as i, the color will appear bright, and conversely. We render the scatterplot by drawing radial splats of R pixels radius, textured with color and luminance computed as above, and using a opacity (alpha) varying from fully opaque in the center to slightly transparent at the borders, to smoothly blend neighbor splats. Setting R is discussed further in Sec. 5.5.

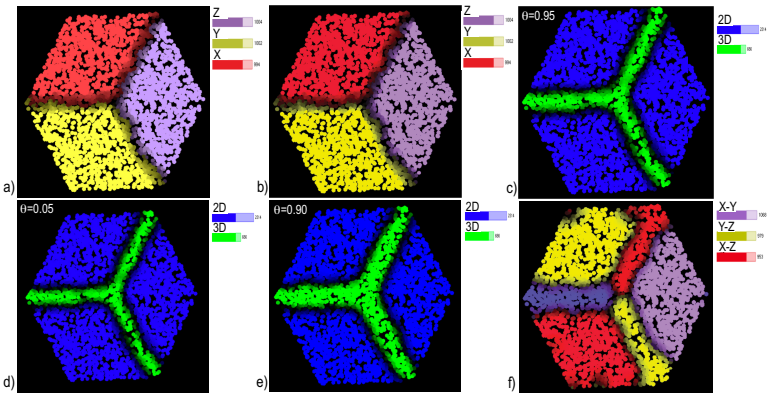


Figure 5.1: Explanatory techniques illustrated on a synthetic cube dataset. The (a) dimension contribution and (b) variance color points by the dimension (X, Y, Z) that makes them most similar to their neighbors. The local dimensionality with total (c), minimal (d), and variance ratio (e) color points by their local intrinsic dimensionality (2D or 3D). The (f) dimensions correlation colors points to indicate the strongest-correlated dimension pair (X-Y, Y-Z, X-Z) close to each point. Bars in the legends show the number of points explained by each dimension (a,b), dimensionality (c,d,e), and dimension pair (f). See Sec. 5.3.

Figure 5.1a,b show a 3K point dataset spread over three faces of an axis-aligned cube (with added noise), projected with PCA to 2D, explained by dimension contribution, respectively variance. Points on each cube face share very similar values of a dimension, so are bright and colored by the respective dimension. It is important to see that these are the original data dimensions (x, y, z), and not latent dimensions synthesized by PCA (eigenvectors). Points along cube edges are dark, since two (or three, for the cube corner) dimensions are needed to explain their similarity with neighbors. Hence, their color coding in the visualization and corresponding legend. Although these two explanations are practically identical for the cube dataset, we will see later on that they can subtly differ, thus both bringing in added value in the projection understanding process. We note that the similarity of the dimension contribution and variance explanations for a similar synthetic

cube dataset was observed originally by Da Silva *et al.* (da Silva et al. (2015)) in their seminal paper where these two techniques were proposed. However, they did not further explore how these explanations vary on real-world datasets and/or how the visual analyst should interpret these differences – a limitation that we aim to alleviate in our current work.

5.3.1 *Adding dimensionality explanation*

Da Silva *et al.*'s explanations (Eqns. 5.2 and 5.3) cannot provide full insights into the structure of high-dimensional data. Take *e.g.* a non-axis-aligned cube like in Fig. 5.1a and embed it into a high-dimensional space. While the data structure stays the same, both distance contributions and variances cannot select a single dimension to explain the cube's faces, since all dimensions contribute to the data structure.

We improve this by explaining the data's *local* (or intrinsic) *dimensionality*. For each neighborhood μ_i of a point $\mathbf{x}_i \in D$, we compute the n eigenvalues α_i of its covariance matrix, sorted decreasingly. From these, we compute the local dimensionality δ of μ_i and its confidence κ in three different ways (see Tab. 1).

Definition	**Dimensionality** δ	**Confidence** κ
Total variance	$\min \delta \Big\vert \frac{\sum_{i=1}^{\delta} \alpha_i}{\sum_{i=1}^{n} \alpha_i} \geq \theta$	$1 - \frac{\sum_{i=1}^{\delta} \alpha_i - \overline{\alpha}}{\sum_{i=1}^{n} \alpha_i}$
Minimal variance	$\left\vert \left\{ \frac{\alpha_i}{\sum_{j=1}^{n} \alpha_j} \geq \theta, 1 \leq i \leq n \right\} \right\vert$	$\frac{\sum_{i=1}^{\delta} \alpha_i}{\sum_{i=1}^{n} \alpha_i}$
Variance ratio	$1 + \min \delta \Big\vert \frac{\sum_{i=1}^{\delta} \Delta\lambda_i}{\sum_{i=1}^{n} \Delta\lambda_i} \geq \theta$	$1 - \frac{\sum_{i=1}^{\delta} \Delta\lambda_i}{\sum_{i=1}^{n} \Delta\lambda_i}$

Table 1: Definitions of local dimensionality and confidence.

Total variance (TV): We define dimensionality δ as the minimal number of largest eigenvalues $\alpha_1 \geq \ldots \geq \alpha_\delta$ needed to explain a user-set fraction θ of the data variance in μ_i. The confidence κ equals how much the sum of these largest δ eigenvalues deviates from the mean of all n eigenvalues.

Minimal variance (MV): The TV model works well when eigenvalues significantly drop. However, take the (limit) case where all eigenvalues are equal. TV then computes $\delta = \theta/n$, even though locally the data is truly n-dimensional. To capture this, we define δ as the number of eigenvalues larger than a minimal user-set variance θ, and confidence κ as the sum of these divided by TV, similar to Kaiser's criterion used in explanatory factor analysis (Cliff, 1988; Jolliffe, 2002).

Variance ratio (VR): Several metrics are known in 3D diffusion tensor analysis to describe the shape of local neighborhoods (O'Donnell and Westin, 2011). We generalize these to nD data and compute dimensionality δ by summing differences of consecutive eigenvalues $\Delta\lambda_i = \lambda_i - \lambda_{i+1}$ normalized by the largest one, λ_1. Each difference captures a significant 'drop' in consecutive eigenvalues, and the sum accounts for the effect of all drops. Thresholding this sum by a user-set θ yields the local dimensionality. P and Falguerolles (1993) and North et al. (1982) have described similar models of local dimensionality. Note that, in the definition of δ for VR (Tab. 1), we define $\lambda_{n+1} = 0$. Also, if $\lambda_1 > \theta$, we set $\delta = 1$; if $\lambda_1 < \theta$, we set $\delta = 0$ (the whole dataset is concentrated in a single n-dimensional point).

Figures 5.1c-e show the total, minimal, and variance ratio explanations for the noisy cube. The thresholds θ are listed in the figure and discussed next in Sec. 5.5. The explanations are color-coded on the projection points, as detailed in the legends. The legend bars' sizes tell how many points are assigned a given explanation (dimensionality). The cube's faces are blue, meaning that these points are locally in $\delta = 2$ dimensional neighborhoods embedded in nD. Close and on the cube edges, green tells that $\delta = 3$ dimensions are needed to explain the data here. The blue and green area are separated by (thin) dark bands, indicating projection areas where the confidence of assigning a dimensionality of $\delta = 2$ or $\delta = 3$ is low – these are the transition areas between the blue ($\delta = 2$) and green ($\delta = 3$) areas. The three local dimensionality explanations are very similar to each other, indicating that the PCA-based analysis underlying all three computations makes sense. For more complex datasets, the explanations can slightly differ and convey interesting insights, similar to the differences between the distance contribution and variance explanations discussed earlier (see Fig. 5.1a,b).

5.3.2 *Adding correlation explanation*

High-dimensional data is often explained by how its dimensions *correlate*. Yet, assessing *global* correlation over an entire dataset is of limited value when the underlying phenomenon is a mix of local (linear) patterns. To address this, we compute and depict correlations over neighborhoods. For each point neighborhood μ_i, we compute the $K = n(n+1)/2$ Pearson or Spearman correlations between all dimension-pairs $(j,k) \in [\![1,n]\!] \times [\![1,n]\!]$. We sort these pairs in descending correlation-strength order, and select the C top-ranked pairs that are most frequent over all points i. This resembles selecting the explaining dimensions in da Silva et al. (2015), but now we select dimension-pairs rather than individual dimensions. We show these C pairs via a categorical colormap, using luminance to map the absolute correlation values. Figure 5.1f shows this for the noisy cube. The legend tells that the three faces map to strong correlations of the three dimensions x, y, and z,

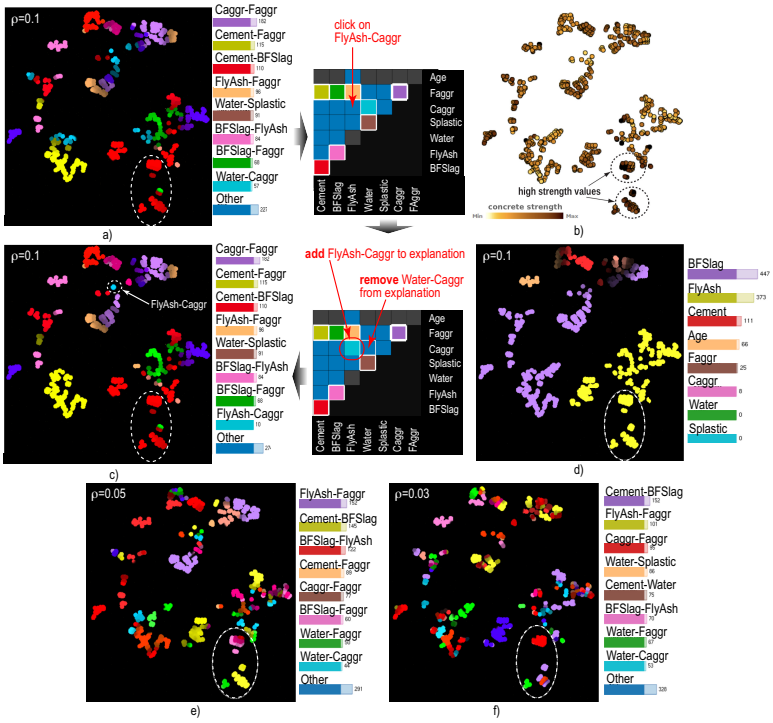


Figure 5.2: Matrix view, *concrete* dataset. Clicking on the *FlyAsh-Caggr* cell (a) allocates a color to it, showing where in the t-SNE projection these two variables are strongly correlated. To make room for this, the weakest-correlated pair *Water-Caggr* is removed from the explanation (c) Additional insight is obtained by color-coding the dependent dimension (c), the variance explanation (d), and correlation views using smaller neighborhood sizes ρ (e,f). See Sec. 5.3.3.

as expected. The edges orthogonal to faces show the same correlation. Indeed, for the face xy, for instance, the orthogonal edge has near-constant x and y, and strongly varying z, values, so x and y are correlated along it.

This visualization can only show the C top-ranked, most frequent, correlations from all possible K ones. However, users may want to examine the presence (or absence) of *specific* correlations. For this, we show the entire set of K dimension-pairs using a *matrix view*. To illustrate how this works, we consider next a real, non-synthetic, dataset example.

5.3.3 *Concrete dataset*

This dataset (Yeh, 1998a; Lichman, 2013) has 1030 samples measuring how 8 ingredients influence concrete strength. The independent dimensions are *Cement*, blast furnace slag (*BFSlag*), fly ash water (*FlyAsh*), superplasticizer (*Splastic*), coarse aggregate (*Caggr*), fine aggregate (*Faggr*), and *Water*, each in kg per cubic meters; and the concrete *Age*, measured in days. One is interested to understand which independent dimensions influence concrete strength.

Figure 5.2a shows the matrix view next to the t-SNE projection of this dataset. Matrix cells are colored by the same colormap as used in the projection. Dark blue tells all dimension-pairs whose correlations have a frequency higher than zero but lower than the C top-ranked pairs. To see where, on the projection, a pair correlates, the user clicks a dark blue cell, *e.g.* the *FlyAsh-Caggr* one in Fig. 5.2a. Note that, for lack of visual space, we did not list all variables in the horizontal and vertical legends of that tfigure. The color used for the C^{th} top dimension-pair (*Water-Caggr*, cyan) is then used for the clicked pair and the C^{th} pair is made dark blue. Doing this shows a single cyan spot in the projection (Fig. 5.2b, dashed circle) – the only place where *FlyAsh* and *Caggr* strongly correlate.

The matrix view supports two other tasks. The cells of the top C (strongest correlated) dimension-pairs are outlined in white, helping one to easily return to the original color mapping after having selected some other dimension-pairs to explain. Rows and columns having many cells with the non-default (dark blue) color indicate *groups* of strongly correlated variables. For instance, the second top row in Fig. 5.2a, for the *Faggr* dimension, shows four such cells that indicate *Faggr*'s strong correlation with *Cement* (yellow), *BFSlag* (green), *FlyAsh* (orange), and *Caggr* (purple), respectively.

Da Silva also used this dataset (da Silva, 2016), also projected with t-SNE, to find attributes that predict high concrete strength. For this, they colored the projection by each of the 8 independent dimensions, and next by the dependent dimension (concrete strength). Figure 5.2b (same as Fig. 5.10 in da Silva (2016)) shows the dependent dimension, allowing one to find two high-concrete-strength clusters. By manually comparing the values of all independent dimensions over these clusters, Da Silva found that *BFSlag* also had high values in these areas. However, this manual comparison of color-coded dimensions is quite tedious.

We next show how our explanatory views help refining the above insights. In Fig. 5.2a,c, we see a correlation between *Cement* and *BFSlag* attributes in the selected region. Now, if *cement* and *BFSlag* correlate with each other, and *BFSlag* correlates with high concrete strength, *cement* likely correlates to concrete strength as well. To search for additional correlations over subsets of points in the selected region (smaller neighborhoods), we next decrease the radius ρ used to compute the

correlation view. In Fig. 5.2e, computed with $\rho = 0.05$, we see a *BFSlag-Faggr* correlation (pink upper cluster), and also a *Water-Faggr* correlation (green lower cluster). Also, the *cement-BFSlag* correlation stays strong in the middle (yellow) cluster. In Fig. 5.2f, computed with $\rho = 0.03$, we see the *cement-BFSlag* and *water-Faggr* correlations in the purple, respectively green, clusters; the red upper cluster shows an additional *Caggr-Faggr* correlation. Now, because *BFSlag* was found to correlate with *Faggr* in this region, *Faggr* might be related to high concrete strength (especially in combination with large *BFSlag* values). And because *Faggr* might be correlated, and we found a *water-Faggr* correlation and a *Caggr-Faggr* correlation, both *water* and *Caggr* might explain high concrete strength.

We now use the variance view (Fig. 5.2d) to get extra insights in the selected region. The entire region is yellow, *i.e.*, points there have a small *FlyAsh* variance. Also, *FlyAsh* varies little also *far beyond* the region borders. Putting it all together: *BFSlag*, *cement*, *Faggr*, *water*, and *Caggr* (but not *FlyAsh*) might together help shaping a regressive model for high concrete strength. Wu et al. (2010) independently studied this dataset for of predictive modeling, showing the Pearson correlation coefficients between the data attributes (Table II in Wu et al. (2010)). They found a relatively strong positive *cement-BFSlag* correlation (0.29), inverse correlations of *BFSlag-Caggr* (-0.31) and *BFSlag-Faggr* (-0.31), and an inverse *Faggr-water* correlation (-0.44). Our findings, obtained via our correlation views, are consistent with these results – except that we do not visualize the *sign* of the correlation.

5.3.4 *Parameters*

Our explanations depend on the following user parameters:

Neighborhood size: Given as a fraction of the projection size (so $\rho \in [0, 1]$), ρ tells the *scale* of the visual structures we want to explain. Figure 5.4 illustrates this for the variance explanation of the *wine* dataset. Smaller ρ values explain finer-grained structures, but can create noisy visualizations, since, in the limit, every (small) neighborhood can be potentially best explained by a different dimension; since we usually do not have as many categorical colors as the dataset's number of dimensions n, many such neighborhoods will not receive an explanation (see Sec. 5.3). Large ρ values will attempt to explain large visual structures by a single dimension, which, in the limit, when ρ equals the projection's size, amounts to showing the dimension having globally least variance, which is not insightful. Good values for ρ range around 0.1 of the projection's size. This is the default value used in all the views in this paper unless otherwise specified. Indeed, for a dataset having a few thousand samples, this ρ value yields a few tens of samples per neighborhood v_i, which is sufficient, as a lower bound,

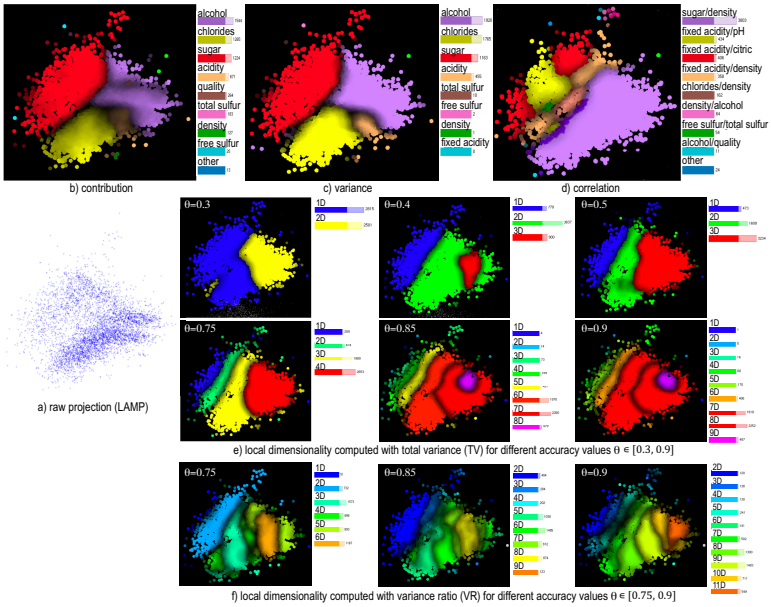


Figure 5.3: Explanation of *wine* dataset. The contribution and variance views (b,c) split the projection in four main clusters, characterized by similar values of sugar (red), chlorides (yellow), and alcohol (purple). The correlation view (d) further explains the yellow and red clusters by the correlation of sugar with density (similar interpretations exist for the red cluster). The dimensionality views (e,f) indicates that the blue area, which falls inside the red zone in (b,c), can be explained by a single dimension, which is thus the earlier-identified sugar dimension. See Sec. 5.4.1.

to reliably compute all the proposed explanations.

Dimensionality threshold: The value $\theta \in [0, 1]$ (Tab. 1) specifies how much of the data's local dimensionality we want to explain. For TV and VR, a high θ value explains more of the local dimensionality, but can lead to projections where most points are marked as high-dimensional, which is not very useful. A too low θ value can generate false confidence that the 2D projection captures all the intrinsic dimensionality of the data. For MV, θ behaves oppositely – low values explain more of the intrinsic data dimensionality. We empirically found that $\theta \in [0.6, 0.9]$ (for TV and VR), respectively $\theta \in [0.05, 0.1]$ (for MV) yield an informative, but not too strict, visualization.

Splat radius: The value R gives the size, in pixels, of the splats that render the explanation and its confidence (Sec. 5.3). Small R values create discrete-looking scatterplots, where the colors of neighbor points

do not visually merge, thereby breaking the color-and-luminance gradients which are key to explaining *regions* in the scatterplot. High R values create too much overlap between neighbor points, so regions smaller than R cannot be visually distinguished. R and the neighborhood radius ρ act as dual scale parameters – ρ controls the scale at which we *compute* explanations, and R controls the scale at which we *render* them. We studied several options of setting R automatically, *e.g.*, based on the average local density of scatterplot points, following similar work in Martins et al. (2014). We found such automatic methods risky, as they tend to indiscriminately 'fill in' gaps of all sizes in a projection, including those which separate faraway point clusters. Hence, we leave R as a parameter for the user to set. A good preset for R is the average distance-to-the-closest-neighbor in the projection, which amounts to $\rho \in [0.03, 0.05]$ of the image size for the figures in this paper.

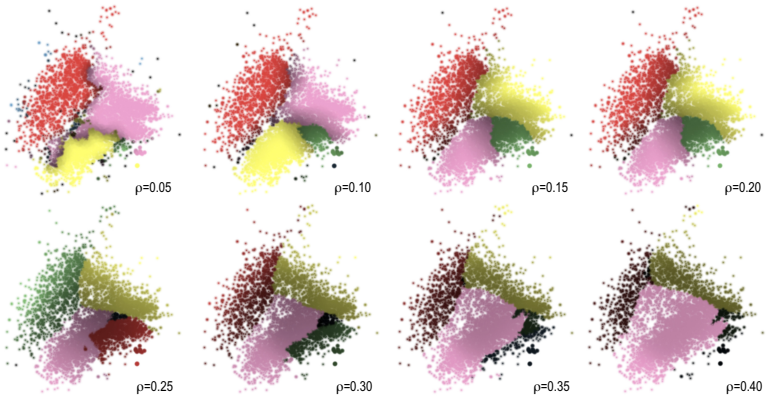


Figure 5.4: Variance explanation for the *wine* dataset, projected by LAMP, for eight values of ρ (as fraction of the projection size). *rho* functions as a scale parameter: As it increases, the computed explanation becomes coarser, and small-scale details are removed. See Sec. 5.4.1.

5.4 APPLICATIONS

We show next how the six explanatory views – distance contribution, variance, correlation, and local dimensionality computed by total variance, minimal variance, and variance ratio – can be combined to extract insights from four non-synthetic datasets. We also correlate these insights with ground truth extracted by independent research that studied the same datasets.

5.4.1 *Wine quality dataset*

We first consider the *wine* dataset, which has 6497 samples of Portuguese *vinho verde* (Cortez et al., 2009a), each with $n = 12$ physicochemical attributes such as acidity, residual sugar, and alcohol rate. Figure 5.3a shows the raw projection of this dataset using LAMP (Joia et al., 2011). Besides a dense point cluster bottom-right, there is not much else this image tells us. While other projection methods, *e.g.* t-SNE, may show better separated clusters, the question still remains how to explain these.

Figures 5.3b-c show the contribution and variance explanations respectively. These are quite similar and split the projection roughly into four areas, explained by small variations of alcohol (purple), chlorides (yellow), sugar (red), and acidity (beige), respectively. The correlation view (Fig. 5.3d) brings additional insights: We see a large purple area bottom-right that matches well the area earlier explained by small variations of chlorides, alcohol, and acidity. Over this purple area, the legend of image (d) tells that sugar and density strongly correlate. Also, we see that the red area in Figs. 5.3b-c, where sugar has a low variation, is now roughly split in Fig. 5.3d into smaller areas – red (fixed acidity-citric acid correlation), yellow (fixed acidity-pH correlation), beige (fixed acidity-density correlation), and brown (chlorides-density correlation). Note that the contribution-variance and correlation explanations are *complementary*: They cannot, when taken separately, split the projection into fine-grained local explanations, but do so when *combined*. Indeed, the red area in Figs. 5.3b-c is further split (explained) by using correlation, as explained above; conversely, the purple area in Fig. 5.3d is further split (explained) by using contribution or variance.

At this point, the analyst may wonder which projection areas are *sufficiently* explained by the above views. The dimensionality view helps here. Figure 5.3e shows the local dimensionality of the projected data, computed by total variance (Sec. 5.3.1). We see how increasingly more dimensions are needed to capture increasing fractions $\theta \in [0.3, 0.9]$ of the total variance – in the limit, we need all $n = 12$ dimensions to explain $\theta = 100\%$ of the variance. More interestingly, we see in Fig. 5.3e a gradient of local dimensionality, from highest in the bottom-right area (red-purple colors for $\theta \geq 0.85$) to blue in the top-left area (blue for $\theta \leq 0.75$). Besides color hue, the local dimensionality gradient is also visible in the brightness, which tells that the confidence κ shows that the color-coded number of dimensions locally explain θ percent of the variance. The effect is very similar to the enridged contour maps used to visualize scalar fields (van Wijk and Telea, 2001): The visual nesting of the 'cushions' created by varying brightness conveys the absolute value of the encoded signal, *i.e.*, the local dimensionality. The way we compute these cushions (Sec. 5.3.1) is, however, completely different to van Wijk and Telea (2001).

The local dimensionality view helps interpreting the contribution, variance, and correlation views as follows: As we have seen, local dimensionality is high in the bottom-right (red-purple) area, where we need 7 to 9 dimensions to explain $\theta = 0.85$ of the data variance. In this area, the contribution-variance and correlation views jointly give us information about only *five* variables – alcohol, chlorides, acidity, sugar, and density. Hence, these two views do not *fully* explain this area, so we need to search for more explanations here. In contrast, the local dimensionality is low in the top-left (blue) area, where we can explain $\theta = 0.75$ of the data variance by a single dimension. From the contribution-variance views, we see that this area is well explained by a small variance of sugar. Hence, in this area, sugar's low variance is sufficient to explain the data.

Figure 5.3f shows the local dimensionality computed by VR as opposed to TV (Fig. 5.3e, for the three largest θ values. While the exact borders of the explained regions differ, we see overall the same pattern, *i.e.*, low dimensionality to the left, respectively high dimensionality to the right, of the projection. The insights described above – obtained with TV dimensionality – stay the same. The actual dimensions assigned to comparable regions in the two explanations are similar – for instance, the blue areas in Fig. 5.3f ($\theta = 0.75$), of local dimensionality 1 and 2, match well the blue-and-green areas in Fig. 5.3e ($\theta = 0.75$) which are also of dimensionality 1 and 2.

Beh and Holdsworth (2012) studied this dataset by correspondence analysis, multiple regression analysis, classification, and visual evaluations. Using the classification technique of Cortez et al. (2009a), they examined the mean value of each attribute for the classification as scored by assessors. They found a relationship between low sugar, density, fixed acidity and volatile acidity, and higher-quality white wine. Also, stronger values of alcohol, pH and sulfur are implied to lead to higher-quality wine. For red wine, high levels of alcohol and sulfur are also found to be a strong quality indicator, while low chloride levels can lead to higher quality red wine. Residual sugar and density are found to be statistically irrelevant in predicting red wine quality. If we compare Fig. 5.3 to these findings, checking for value ranges by brushing the projection, we find several matches: The high-quality wines (brown area, Fig. 5.3b) have indeed high sulfur (brown area, Fig. 5.3c) and are in a region of high sugar-density correlation (both these attributes having low values, confirmed by brushing – purple area, Fig. 5.3c). We confirm the additional layer behind sugar-density correlation (purple area, Fig. 5.3c), specifically in regions where similarity is explained by chlorides and alcohol (purple and yellow areas, Figs. 5.3b,c), as all these attributes add to predicting wine quality. In the purple area in Fig. 5.3c, the sugar-density correlation is roughly of 0.9. This is in line with the sugar-density correlation of 0.83 reported for *all* the samples of this dataset by earlier studies (Zeng, 2021).

5.4.2 *Software quality dataset*

This dataset contains 6773 software projects from SourceForge written in the C programming language Meirelles et al. (2010). Each project has 10 independent dimensions, these being metrics used in software engineering to gauge software quality: coupling between modules, complexity, lack of cohesion, number of source files, number of lines of code, number of function parameters, number of public variables, number of methods, number of data members, and structural complexity. Two additional dimensions measure the number of downloads and number of developers of a given software project.

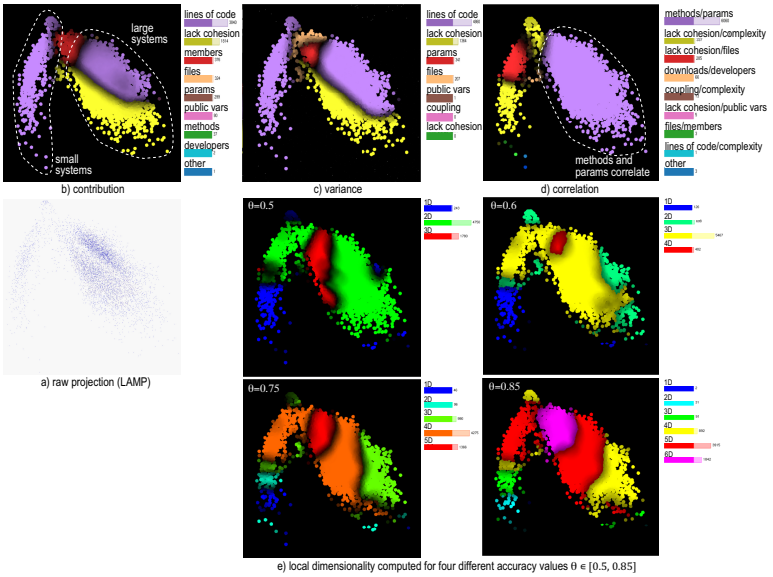


Figure 5.5: Explanation of *software* dataset. The contribution (b) and variance (c) views show two purple lobe-like clusters corresponding to small, respectively large, systems. The correlation view (d) shows that large systems also have their method and parameter counts correlated. The local dimensionality views (e) shows that the two lobes can be explained by about three dimensions, while the area connecting them requires more effort to explain. See Sec. 5.4.2.

Figure 5.5a shows the dataset projected with LAMP. As for the wine dataset (Sec. 5.4.1), the raw projection is not very informative. Figures 5.5b,c show the projection explained by contribution, respectively variance. As for the wine dataset, these two explanations are very similar: The purple and yellow regions in both Fig. 5.5b,c show software systems which are mostly similar due to size (lines of code), respectively complexity. The two *disjoint* purple regions indicate two groups of systems which are similar due to two different *value ranges* of lines of

code. Brushing the image shows that projection is roughly split into a left lobe consisting of small software systems, and a right lobe containing large systems. However, the contribution and variance explanations are not *identical*: The red region in Fig. 5.5b shows systems which are similar in number of members. This region matches very well the union of the red and beige regions in the variance explanation (Fig. 5.5c), *i.e.*, systems with similar number of parameters or files. Hence, the number of members, parameters, and files appear to be correlated in this region.

The correlation view (Fig. 5.5d) adds more insights: The large purple area indicates systems which have correlated numbers of methods and parameters. From the earlier correlation/variance analysis, we know that these are large systems. Upon further study of the names of these systems in the original data (Meirelles et al., 2010), we find that these are mainly software libraries – for which, indeed, the total number of methods and total parameter count are correlated, since, in libraries (APIs), methods have typically similar parameter counts. The left lobe of the projection, *i.e.*, the small software systems, are yellow and red, indicating correlated lack-of-cohesion and complexity, respectively correlated lack-of-cohesion and number of files. Like for the wine dataset, such findings are only possible when joining the three different explanatory views. The correlated lack-of-cohesion with complexity is also a known signal in software quality analysis: Poor quality software is very often incohesive *and* complex (Richter, 1999).

We now examine the dimensionality of the projected data. Figure 5.5e shows this for four different values of θ. Overall, these views tell us that the extremities of the two projection lobes are quite low-dimensional, being well explained by about three dimensions. In contrast, the area connecting the lobes requires five to six dimensions to explain. This area roughly corresponds to the red, respectively red-and-beige, regions in the contribution, respectively variance, views. The dimensionality view tells us that more explanations are needed in this central area since the projection is there not sufficiently well explained by the number of members, respectively lack-of-cohesion and number of parameters dimensions.

We next compare our findings with those of Meirelles et al. (2010). They found high correlations of complexity *vs* lack of cohesion (Pearson: 0.786, the highest correlation of all dataset dimension-pairs; Spearman: 0.773; Kendall tau: 0.597); and number of methods *vs* parameters (Pearson: 0.762; Spearman: 0.765; Kendall tau: 0.596). They also found a strong correlation between complexity and lines of code (Pearson: 0.666; Spearman: 0.685; Kendall tau: 0.497), the third strongest correlation for complexity, and a correlation between lack of cohesion and lines of code (Pearson: 0.472; Spearman: 0.490; Kendall tau: 0.341), the second strongest for the lack-of-cohesion attribute. These two correlations combined match our finding of complexity and lack of cohesion correlated (Fig. 5.5d, yellow areas) over a region of similar lines-of-code

values (Fig. 5.5b, left purple lobe). Their strong-reported correlation of number of methods *vs* number of parameters noted above matches the purple lobe in Fig. 5.5d, on which we found a correlation of roughly 0.92. Note that the findings of Meirelles *et al.* are *averages* over the entire dataset. Our correlation view refines such insights by showing *local* correlations over subsets of the data.

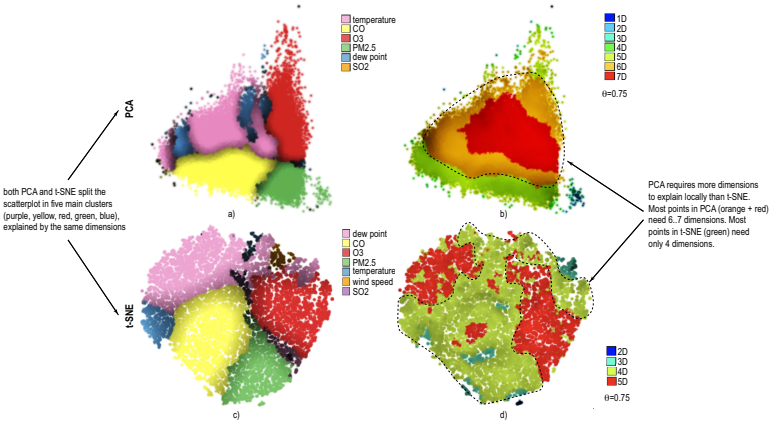


Figure 5.6: Explanation of *city pollution* data, PCA and t-SNE projections. The variance views (a,c) show that both projections split the data into clusters with similar explanations. The dimensionality views show that PCA needs more additional dimensions to explain its clusters (b) than t-SNE (d). See Sec. 5.4.3.

5.4.3 *City pollution dataset*

This dataset, from the UCI Machine Learning repository, contains 420768 measurements of 6 air pollutants (PM2.5, PM10, SO2, NO2, CO, O3) and 6 meteorological variables (temperature, pressure, dew point temperature, rain, wind direction, and wind speed) measured hourly from March 2013 to February 2017 at 12 sites in Beijing (Zhang et al., 2017). We removed the time dimension (aggregating all measurements together) and projected the resulting dataset using both PCA and t-SNE.

We use this dataset to contrast how our explanations work for different projection types. Figure 5.6a shows the variance explanation for PCA. This projection is split into four similar-size regions explained by the temperature, CO, O3, and PM2.5 dimensions. The dimensionality explanation of the PCA projection (Fig. 5.6b, $\theta = 0.75$) shows that we need five to seven dimensions to explain the projection, with more dimensions needed in the center thereof. The t-SNE projection is also split into similar-variance zones explained by the same variables (temperature, CO, O3, and PM2.5). Interestingly, these regions are placed relatively

to each other quite similarly to their counterparts in the PCA projection. The dimensionality explanation of the t-SNE projection (Fig. 5.6d, $\theta = 0.75$) is very different from PCA's one: We do not see the low-to-high dimensionality gradient present in Fig. 5.6b; rather, the projection is locally either 4-dimensional (green) or 5-dimensional (red). Hence, t-SNE achieves a better 'spread' of the high-dimensional dataset in 2D than PCA. More interestingly, the red-green borders in Fig. 5.6a match relatively well the borders of the red and pink regions in Fig. 5.6c. This tells us that the dew-point and O3 explained regions in that figure are five-dimensional, whereas the CO, PM2.5, and temperature explained regions are four-dimensional, respectively.

5.4.4 *Air quality dataset*

This dataset, also from the UCI repository, has 9358 samples of air quality measurements (CO, NOx, NO2, benzene, and non-metanic hydrocarbons (NMHC)) done by both an experimental sensor and a reference ground-truth (GT) analyzer. Apart from these, temperature, relative humidity (RH) and absolute humidity (AH) are measured. Data were recorded from March 2004 to February 2005 in a highly polluted area of an Italian city (Vito et al., 2008), and its authors outline significant differences between the experimental sensor and GT values.

As for the city pollution dataset, we use our views to explain the PCA and t-SNE projection of this data (aggregating the time dimension). Figure 5.7a shows the variance explanation of the PCA projection. This projection shows five visually separable clusters (dashed outlines A-E). Cluster D is actually an overlap of three clusters explained by the dimensions CO(GT) – pink, AH – yellow, and NMHC (GT) – red. The dimensionality view (Fig. 5.7b, $\theta = 0.68$) increases the confidence in the variance explanation: Clusters A, B, and C, which showed little overlap of explanations, are intrinsically two-dimensional, so we can trust the PCA projection here. Cluster E, which has a line structure, is intrinsically one-dimensional, so its explanation by the single dimension NOx (GT) in Fig. 5.7a is complete. In contrast, cluster D is two-to-three dimensional, which is exactly what its explanation by three 'overlapping' dimensions in Fig. 5.7a tells us. Figure 5.7c shows the variance explanation of the t-SNE projection. We see here six visually distinct clusters (A′-F′). Upon closer inspection, by brushing, we found that A′ corresponds roughly to the union of A, B, and the pink part of D; B′ corresponds to the red part of D; D′ and F′ correspond to the yellow part of D; C′ corresponds to C; and E′ corresponds to E. Saliently, the colors in Fig. 5.7c correspond almost perfectly to visually distinct clusters. We also see no dark points in this figure, meaning that the confidence of the explanation is very high. Hence, the t-SNE projection both *groups* similar-value points better than PCA (see the pink points), and *separates* different-value points better (see the red, yellow, and green points). The

dimensionality view (Fig. 5.7d, $\theta = 0.68$) confirms this: except a tiny red area, all points indicate neighborhoods of intrinsic dimensionality of one (blue) or two (green). Since this is a 2D projection, this tells us that t-SNE did a very good job in preserving the high-dimensional data structure, and in any case, better than PCA.

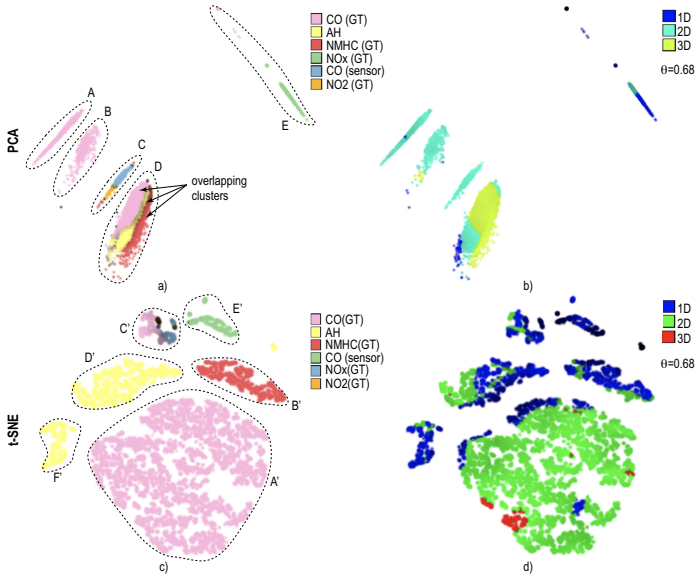


Figure 5.7: Explanation of *air quality* dataset, PCA and t-SNE projections. Colors in the variance views (a,c) help finding the main variable explaining what makes points in a cluster similar. The local dimensionality views (b,d) tell us how many extra variables we need to fully explain these clusters. See Sec. 5.4.4.

5.5 DISCUSSION

We detail several aspects of our method, as follows.

Genericity and scalability: Our method can handle any type of quantitative data projected by any MP technique. Correlations and PCA are computed with Eigen (Jacob and Guennebaud, 2020). Since explanations are computed and rendered independently on local point neighborhoods, we parallelized this using multithreading on the CPU. We generated all images in this paper in seconds for datasets up to tens of thousands of points, tens of dimensions, on a modern PC (3.6 Ghz CPU, GeForce 900 GPU). Table 2 shows timing measurements for several datasets having a wide range of dimensions n, samples N, and sizes ρ of the neighborhoods v_i, sorted ascendingly on the total

attribute count $n \cdot N$.

Table 2: Computational performance of explanatory views

Dataset	Dimensions	Samples	Total	Time (secs)		
	n	N	$n \cdot N$	$\rho = 0.1$	$\rho = 0.2$	$\rho = 0.3$
D1	17	143	2431	0.013	0.016	0.016
D2	20	740	14800	0.025	0.028	0.029
D3	32	520	16640	0.016	0.019	0.019
D4	11	4177	45947	00.68	0.069	0.082
D5	25	2584	64600	0.046	0.045	0.047
D6	11	6497	71467	0.133	0.136	0.165
D7	179	11500	2058500	2.611	3.168	5.033
D8	64	41188	2636032	0.845	3.082	13.884

Combining explanations: The examples in Secs. 5.3 and 5.4 show that no single explanation suffices. One has to combine the partial insights of different explanations from the total six ones (distance contribution, variance, three local dimensionality variants, and dimensions correlation) to arrive at relevant, stronger, findings. In this process, one can (a) use explanations of the *same* type, *e.g.* local dimensionality, which, where matching, strengthen the obtained findings; or (b) explanations of *different* types, *e.g.* correlation and variance, which performs 'logical AND' like operations on their partial insights.

Projection quality: Our explanations rely on the assumption that points close in $P(D)$ correspond to points close in D – that is, that the projection exhibits high values of trustworthiness (Venna and Kaski, 2006b). In other words, our explanations require that *the neighborhoods shown in a projection are meaningful.* If they are, then we can explain them. If not, then we will produce wrong explanations, but arguably *any* use of such a projection will be flawed, not only our explanations, since the projection contains errors. The extent to which various MP techniques realize this neighborhood preservation varies (Espadoto et al., 2019). One way to address this is to use projection error views (Martins et al., 2014) to exclude neighborhoods which do not respect this condition (Rodrigues et al., 2019), or refine their computation by *e.g.* using larger radii ρ. To address this issue, Table 3 shows the continuity, trustworthiness, and Shepard correlation quality metrics computed for all the datasets and all the projections discussed earlier in this paper. For the exact definitions of these metrics, we refer, for brevity, to Table 5 in Espadoto et al. (2019).

Dataset	Projection	Continuity	Trustworthiness	Shepard
Concrete (Fig. 5.2)	t-SNE	0.99810535	0.99517108	0.53527163
Wine (Figs. 5.3,5.4)	LAMP	0.84132354	0.92384026	0.79137224
Software (Fig. 5.5)	LAMP	0.90646675	0.98470294	0.91487058
City pollution (Fig. 5.6)	PCA t-SNE	0.93095898 0.99888747	0.99232401 0.98818749	0.95164997 0.84766134
Air quality (Fig. 5.7)	PCA t-SNE	0.94080419 0.99916219	0.99208358 0.99601412	0.97113638 0.56614243

Table 3: Quality metrics for all projections and datasets in this paper.

Table 3 shows that all the computed projections are of high quality, their values being very close to the maximum value of 1. For t-SNE, the Shepard correlation is relatively lower, but this is expected, as this metric quantifies the preservation of distances and the t-SNE technique does not aim to preserve distances, but neighborhoods. All in all, the projections shown in this paper are of sufficiently high quality to vouch their visual exploration by means of our explanatory techniques, and also to trust their computation which relies on the assumption of high trustworthiness already mentioned above.

Limitations: While we can *technically* handle datasets of any dimensionality n, we need more variables for the explanation as local dimensionality grows. Also, the correlation is $O(n^2)$ in computation and space needed for the dimension matrix (see Fig. 5.2 and related text). Our method works well up to 20 dimensions in practice; it does not target datasets with hundreds of dimensions such as from deep learning. Yet, such datasets have *abstract* dimensions which do not have a meaning for users, so using them to explain projections is likely not desirable. Our method scales *visually* well even for many dimensions, since it uses only the *top ranked* ones which contribute to explaining most of the projected points (Sec. 5.3).

One can ask whether using nD point neighborhoods $\xi_i = \{\mathbf{x} \in D | \ \|\mathbf{x} - \mathbf{x}_i\| \leq \rho\}$, $P(\mathbf{x}_i) = \mathbf{y}_i$, instead of 2D neighborhoods ν_i (and their correspondents μ_i in nD), is a valid option. Doing this is technically trivial, but we argue against it: We aim to explain the point-groups one sees in a *projection* (2D scatterplot) and not the point-clusters that exist in nD, but may *not* be visible in 2D due to *e.g.* projection continuity issues Venna and Kaski (2006a). Also, setting the neighborhood size ρ would be tricky for ξ_i, as one has to assess what is the 'natural' scale of patterns in nD. This motivates our choice to use 2D neighborhoods as a basis for our explanations.

A separate limitation involves color coding, which is used to create categorical color maps (contribution, variance, and correlation plots) and also ordered color maps (dimensionality plot). As explained in Sec. 5.4, several such plots are to be used together to arrive at a good understanding of a projection. This may potentially confuse users since the respective colormaps contain similar colors. The problem can be partly alleviated by designing colormaps with a smaller overlap in terms of such colors. However, as we next aim to extend our approach with additional explanatory views, this alleviation strategy is not a full solution. For now, we prominently display the respective color legends next to each explanatory plot, aiming thereby to attract the attention of the user of the particular meaning of colors in that plot.

User perception: As our techniques aim to explain the patterns one sees in a projection, they should be tested in experiments where subjects use them to perform some explanatory tasks. Earlier studies (Etemadpour et al. (2014)) provide good guidelines of perceptual cues and visual tasks that users address with projections. We aim to extend this work by making such tasks more specific to include explanations that refer to the names of involved dimensions. With this set of tasks, we can next present various combinations of datasets D and projections $P(D)$, computed by several projection techniques P to the users, to find which are the dataset and/or projection-technique aspects that best suit our explanatory techniques. A similar study can be used to find optimal parameters for our explanatory techniques.

5.6 CONCLUSIONS

We have presented a set of visualizations for explaining the visual patterns present in 2D projections of high-dimensional data in terms of the underlying data dimensions. We extended the explanations proposed in earlier work (da Silva et al., 2015) by three ways to evaluate the local data dimensionality and a technique to detect and inspect local dimension correlations. We show that the combined visual analysis of all these explanatory techniques can lead to non-trivial insights in the data that correlate well with independent findings obtained using other methods. We illustrate our approach on five experimental datasets. Our methods are simple to use, have a few parameters with good presets and clear effects, and scale well computationally to datasets of hundreds of thousands of samples and 10..20 dimensions.

Several extensions to our work are possible. Adding more explanation types, such as inverse correlation, correlation of more than two dimensions, or the presence of specific nD data patterns, is a low hanging fruit. We aim to compute, in parallel, a wide range of local explanations based on a pattern library, and next show the most salient ones in the final view, thereby enriching the current contribution,

variance, correlation, and dimensionality views. This would perform a scagnostics-like (Wilkinson et al., 2005) local analysis of the projection, but using patterns described by the high-dimensional data rather than by the scatterplot. Computing a hierarchical explanation, where projection regions are recursively split by additional explanations, is another direction we aim to pursue.

6

QUANTITATIVE AND QUALITATIVE COMPARISON OF 2D AND 3D PROJECTION TECHNIQUES FOR HIGH-DIMENSIONAL DATA

Abstract: Projections are well-known techniques that help the visual exploration of high-dimensional data by creating depictions thereof in a low-dimensional space. While projections that target the 2D space have been studied in detail both quantitatively and qualitatively, 3D projections are far less well understood, with authors arguing both for and against the added-value of a third visual dimension. We fill this gap by first presenting a quantitative study that compares 2D and 3D projections along a rich selection of datasets, projection techniques, and quality metrics. To refine these insights, we conduct a qualitative study that compares the preference of users in exploring high-dimensional data using 2D *vs* 3D projections, both without and with visual explanations. Our quantitative and qualitative findings indicate that, in general, 3D projections bring only limited added-value atop of the one provided by their 2D counterparts. However, certain 3D projection techniques can show more structure than their 2D counterparts, and can stimulate users to further exploration. All our datasets, source code, and measurements are made public for ease of replication and extension [1].

6.1 INTRODUCTION

Visual exploration of high-dimensional datasets is a key component of modern data science pipelines, with many applications spanning disciplines as diverse as social sciences, medicine, biology, and the exact sciences (Hoffman and Grinstein, 2002; Liu et al., 2015; Kehrer and Hauser, 2013; Tang et al., 2016). In the last decades, many visualization methods have been proposed for high-dimensional data, such as parallel coordinates (Inselberg and Dimsdale, 1990), table lensing (Rao and Card, 1994; Telea, 2006), and scatterplot matrices (Becker et al., 1996). Dimensionality reduction (DR) methods, also known as projections, occupy a particular place in this palette of methods, as they are able to handle datasets having both a very large number of samples (also called observations or data points) and dimensions (also called attributes or variables). Tens of projection methods have been proposed by the information visualization (infovis) and machine learning (ML) communities (Nonato and Aupetit, 2018; Sorzano et al., 2014; van der Maaten and Postma, 2009),

1 This chapter is based on the paper 'Quantitative and Qualitative Comparison of 2D and 3D Projection Techniques for High-Dimensional Data' (Tian et al., 2021d).

such as the by now famous t-SNE (van der Maaten and Hinton, 2008) technique.

Choosing a *suitable* projection technique for a given context (application, task, or dataset) is critical since, even for the same dataset, different techniques yield different visualizations, thus leading to potentially different insights and courses of action in the underlying problem solving. This issue, well recognized in the infovis and ML communities, has been mainly addressed by surveys that compare projection techniques from various perspectives, including type of algorithm used (Yin, 2007; Sorzano et al., 2014), types of errors generated (Heulot et al., 2017; Nonato and Aupetit, 2018), and types of tasks addressed (Hoffman and Grinstein, 2002; Kehrer and Hauser, 2013). The most recent survey in this area (Espadoto et al., 2019) aimed to provide fine-grained quantitative evidence to help practitioners choose suitable projections by comparing 44 techniques over 18 datasets from the perspective of 7 quality metrics. The study outlined that, from the perspective of such metrics, most algorithms fare relatively similarly, after one optimizes for their various hyperparameters.

All the above work in comparing projection techniques considered only *two-dimensional* variants thereof, which reduce the high-dimensional data to 2D scatterplots. While 2D projections are the most common in practice, 3D projections have also been proposed (Poco et al., 2011; Coimbra et al., 2016; Sedlmair et al., 2013). Some researchers argue for their added value in terms of better capturing the structure of high-dimensional data (Jolliffe, 2002; Poco et al., 2011; Coimbra et al., 2016). Other researchers argue that 3D projections are challenging to use given the need to choose suitable viewpoints and the presence of clutter and occlusion (Newby, 2002; Westerman et al., 2005). However, 3D projections have been far less studied in the infovis literature – to our knowledge, no quantitative studies have measured 3D projections as Espadoto et al. (2019) did for 2D projections.

The effectiveness of projections in explaining data structure can be increased by *explanatory* tools that annotate the scatterplots to highlight the perceived patterns in terms of the underlying data dimensions, as introduced by da Silva et al. (2015). While such tools have been shown to add value when analyzing 2D projections (van Driel et al., 2020; Tian et al., 2021c), whether and how much they support 3D projections has, to our knowledge, not been studied.

In this chapter, we aim to shed more light on how 3D projections fare when compared to their 2D counterparts by the following contributions:

- We run a *quantitative* study that compares 29 projection techniques, run to create both 2D and 3D scatterplots, from the perspective of 3 quality metrics over 8 high-dimensional datasets. We compare the computed quality metrics of the respective 2D and 3D scatterplots to gauge the added-value of the third dimension;

- We perform a *qualitative* user study that compares the resulting 2D and 3D projection scatterplots, augmented with the visual explanation proposed by da Silva et al. (2015), from the perspective of explaining projection patterns by the data dimensions;
- Our two studies show that, in general, 3D projections have roughly the same quality (measured by metrics and user feedback) as compared to their 2D counterparts, while they require more effort to analyze. However, we also found that, in some cases, 3D projections – when augmented by visual explanations – can show more data structure; and they can motivate users to explore the data more than 2D projections do.

This chapter is structured as follows. Section 6.2 introduces several notations and discusses related work on evaluating 2D and 3D projections. Section 6.3 presents our first contribution, the quantitative comparison of 2D and 3D projections. Section 6.4 presents our qualitative study of the same projections, augmented by visual explanations. Section 6.5 discusses the main findings and limitations of our study. We conclude by outlining directions of future work (Sec. 6.6).

6.2 RELATED WORK

6.2.1 *Preliminaries*

We start by reminding the reader about some key concepts and notations, useful for explaining related work as well as our contribution (related notations, albeit not that exhaustive, were introduced in Chapter 2).

Let $\mathbf{x} = (x^1, \ldots, x^n)$, $x^i \in \mathbb{R}, 1 \leq i \leq n$ be a n-dimensional (nD) real-valued sample, and let $D = \{\mathbf{x}_i\}$, $1 \leq i \leq N$ be a dataset of N samples. Let $\mathbf{x}^j = (x_1^j, \ldots, x_N^j)$, $1 \leq j \leq n$ be the j^{th} dimension of D. Thus, D can be seen as a table with N rows (samples) and n columns (dimensions). A projection technique is a function

$$P : \mathbb{R}^n \rightarrow \mathbb{R}^q, \tag{6.1}$$

where $q \ll n$. In our work, we consider $q \in \{2, 3\}$ and denote the corresponding projection functions by P_2, respectively P_3. The projection $P(\mathbf{x})$ of a sample $\mathbf{x} \in D$ is a qD point. Projecting an entire dataset D yields a qD scatterplot, denoted as $P(D)$. The projection function P is also influenced by so-called *hyperparameters* which are typically fine-tuned by the user to optimize for specific quality metrics (discussed below).

The quality of a projection technique P can be gauged by several metrics defined as

$$M : \{(D, P(D))\} \rightarrow \mathbb{R}_+. \tag{6.2}$$

A metric M measures how well the projection $P(D)$ captures specific properties of the dataset D, the underlying idea being that a good projection will keep similar points in D close to each other in $P(D)$. We detail the specific metrics used in our work in Sec. 6.3.3.

6.2.2 *Evaluating projections*

Since Principal Component Analysis (PCA) (Pearson, 1901; Hotelling, 1933) was first proposed, tens of different projection techniques have been developed, offering many options to data scientists, but also the added challenge in choosing a suitable technique for their goals. To guide this choice, several surveys of projection techniques have been performed. We organize these surveys from the perspective of their goal, as follows.

Technique-centric surveys: These works aim to compare projection methods from the viewpoint of the cost function used to create $P(D)$ from D and the algorithms used to optimize this cost. Fodor (2002) presented the first such survey that we are aware of, which organizes 12 projection techniques in a taxonomy based on their respective cost functions. Sorzano et al. (2014) discussed 30 such techniques with a focus on optimization heuristics and cost functions. Cunningham and Ghahramani (2015) refined the work of Sorzano et al. (2014) with a focus on linear projections. Conversely, Yin (2007) performed a survey for nonlinear projections. Engel et al. (2012) proposed a taxonomy covering nine projections from the viewpoint of out-of-sample ability and computational complexity. Bunte et al. (2012) proposed a theoretical framework to unify nine existing projection techniques from the perspective of how similarity is computed and which error metric a projection minimizes. Finally, Xie et al. (2017) surveyed 27 variants of the Random Projection (RP) method (Dasgupta, 2000), aiming to provide a literature guide to this subclass of techniques.

Task-centric surveys: These surveys categorize projection techniques based on the visual exploration tasks that these support. Buja et al. (1996) and Hoffman and Grinstein (2002) compared projections from an interaction perspective. Kehrer and Hauser (2013) compared projections with other visualization algorithms from the perspective of visual exploration of multidimensional, multi-source, and multi-type data. A similar comparison of projections with other visualization algorithms was performed by Liu et al. (2015). Nonato and Aupetit (2018) surveyed the use of 28 projections in visual analytics (VA) tasks, and categorized these based on the type of errors that they produce and their effect on the performed tasks.

Quantitative surveys: These works compare projections by measuring various quality metrics (M, Eqn. 6.2) on different datasets. Gisbrecht and Hammer (2015) evaluated 10 projection techniques on 3 synthetic datasets from the perspective of one quality metric as well as computational complexity. van der Maaten and Postma (2009) evaluated 14 projection techniques from the perspective of three quality metrics, out-of-sample ability, and computational complexity. More recently, Espadoto et al. (2019) presented the most comprehensive, to our knowledge, quantitative evaluation of projections, which included 44 techniques evaluated against 7 quality metrics over 18 datasets. For each technique, grid-search was used to derive optimal hyperparameter values. We use the work of Espadoto et al. (2019) as a model and inspiration for our comparison of 2D with 3D projections (see Sec. 6.3).

The above surveys provide a wealth of information helping practitioners in understanding how different projection techniques operate and how to choose a suitable one for a given problem context. However, they largely omit 3D projections.

6.2.3 *Three-dimensional projections*

Technically, most existing projection techniques can be used equally easily to create a 2D or a 3D projection. Using a 3D projection would be likely advantageous, since there are more dimensions ($q = 3$) which can capture the structure of the high-dimensional data. Yet, the literature on 3D projections is far less rich than on their 2D counterparts.

A first challenge for 3D projections is finding a good way to explain the patterns that one sees in them in terms of the original data variables or dimensions, a problem that is inherent also to 3D scatterplots which are not produced by projection techniques. This can be done by using multiple 2D views linked by interaction (Piringer et al., 2004) or by smoothly animating transitions between 2D scatterplot views (Elmqvist et al., 2008; Sanftmann and Weiskopf, 2009). More specific for 3D projections, Coimbra et al. (2016) proposed a tool to aid users in choosing suitable viewpoints for a 3D projection and interpreting the spread of points along the screen's X, Y, and depth axes. Additional explanatory techniques for 3D scatterplots are discussed in Sec. 2.6.

A second, more fundamental, challenge is to show why, what for, and how much 3D projections are better than 2D projections. There is evidence both for and against 3D projections (Tavanti and Lind, 2001). For visualizing text data, 2D projections were found easier to use and interact with than 3D projections (Newby, 2002; Westerman et al., 2005). Sedlmair et al. (2013) empirically found 2D and 3D projections equally effective at visual cluster separation tasks. 2D projections were found to work better for tasks related to inter-sample distance assessment

and searching specific sample structures (Westerman and Cribbin, 2000; Fabrikant, 2000). On the other hand, Jolliffe (2002) argued that 3D projections are better at encoding data structure for datasets with intrinsic dimensionality exceeding three. Other arguments in favor of 3D scatterplots (as opposed to their 2D counterparts) are better showing variations in sample density (Sanftmann and Weiskopf, 2009) and decrease of information loss (Chan et al., 2014). Yuan et al. (2021) also showed how specific sampling methods can be used to decrease the amount of occlusion in 3D scatterplots while retaining the patterns these visually encode. 3D scatterplots allow one to easier select specific structures, *e.g.*, point clusters for further investigation than corresponding 2D scatterplots, as the third dimension allows more space for getting these structures separated from each other (Yu et al., 2012). A survey of use-cases where 3D scatterplots are preferable to 2D ones was produced by Sanftmann and Weiskopf (2012). The well-known TensorFlow (TensorFlow, 2021) embedding tool features both 2D and 3D projections using UMAP, PCA, and t-SNE, and uses 3D as default view.

Closest to our work, Poco et al. (2011) compared 2D and 3D projections computed using the LSP projection technique (Paulovich et al., 2008). Their quantitative comparison (by a single quality metric) showed higher accuracy for the 3D projection; the qualitative comparison (done by user studies) showed increased user confidence and satisfaction. Another example arguing the decrease of information loss for 3D projections, along the lines first argued by Chan et al. (2014), is given by Coimbra et al. (2016) and also discussed in Sec. 2.5. However, both above papers (Poco et al., 2011; Coimbra et al., 2016) only studied one projection technique, and for this used a single quality metric. Generalizing such findings for more 3D projections needs more evaluations – a task we approach in this chapter.

6.2.4 *Explaining projections*

Whether 2D or 3D, 'raw' projections that show only the scatterplot $P(D)$ are of little use. Hence, several techniques aim to enrich such scatterplots with additional information to help users understand the visual structures they contain. The simplest explanation color codes points in $P(D)$ by the value of a dimension $\mathbf{x}^j$ or, for image data, a thumbnail representing each sample point (Eler et al., 2009). While simple to implement, understanding how *several* such dimensions explain the plot requires the use of small multiples or manually cycling through color-coding all dimensions. Biplot axes (Greenacre, 2010; Gower et al., 2011) and axis legends (Broeksema et al., 2013; Coimbra et al., 2016) explain the projection's global structure in terms of the dataset dimensions. Local projection errors (Aupetit, 2007; Schreck et al., 2010; Martins et al., 2014) explain how well visual patterns in $P(D)$ encode the structure of

the corresponding data in D. A more comprehensive discussion of explanatory techniques is given in Sec. 2.5.

Besides projection errors, local explanations also aim to explain what, in the data, is common to *groups* of close points in $P(D)$, such as the contribution and variance of each dimension $\mathbf{x}^j$ (da Silva et al., 2015); correlation of two dimensions $\mathbf{x}^j$ and $\mathbf{x}^k$, local dimensionality (van Driel et al., 2020); and salient values common to point clusters (Paulovich et al., 2012), to mention only the most common such techniques. The key added value of such techniques is that they *explicitly* annotate visual structures in the projection $P(D)$ with information from D, thereby making it directly visible what these structures mean data-wise. To our knowledge, such local explanations, most notably the ones in (da Silva et al., 2015; van Driel et al., 2020), have not been used so far for 3D projections. Related to our research question, we would like to find out whether 3D projections would fare better than their 2D counterparts when supported by such explanations.

6.3 QUANTITATIVE STUDY

As explained in Sec. 6.2, there is currently very little quantitative evidence on how 3D projections perform, in terms of quality metrics, as compared to their 2D counterpart. We address this problem by designing and evaluating a *benchmark*, similarly to the earlier one proposed by Espadoto et al. (2019) for 2D projections, which we will next refer to as the '2D benchmark' for simplicity. Constructing the benchmark involves selecting a number of datasets (Sec. 6.3.1), projection techniques (Sec. 6.3.2), and quality metrics to compute (Sec. 6.3.3). We describe these next, also outlining important aspects where we differ from the 2D benchmark.

6.3.1 *Datasets*

To compare 3D *vs* 2D projections, we first selected a number of 8 datasets. Table 4 lists their details, including their sparsity ratio $\gamma_n = 1 - \frac{u}{nN}$, $\gamma_n \in [0, 1]$, where u is the number of non-zero data values; and intrinsic dimensionality $\rho_n \in [0, n]$, defined as the number of principal components (of the total n), computed by PCA, needed to explain 95% of the data variance (Espadoto et al., 2019). More information about these datasets is available in the supplementary material. These metrics can be interpreted as follows: The sparsity ratio is typically quite high for text word-vectors (the data is sparse), and quite low for table data having a small number of dimensions (the data is dense). If a dataset is sparser, its points are closer in the nD space (Bellman, 1957; Beyer et al., 1999), and as a consequence projection techniques have more challenges to identify and separate point-clusters in the projec-

tion space. The intrinsic dimensionality intuitively tells how many dimensions we actually need to represent the data. Datasets having an intrinsic dimensionality equal to, or close to, q are far easier to project to qD, as their structure can be 'unfolded' to be mapped to the qD space. This was recognized early on by algorithms such as Isomap (Tenenbaum et al., 2000) which explicitly exploited the (low-dimensional) manifold-like structure of the data when constructing the projection. Conversely, datasets having a high intrinsic dimensionality are far more challenging to project.

The metric values in Tab. 4 show that our selected datasets cover quite different characteristics, in line with those selected in the 2D benchmark. Using more datasets is definitely desirable. However, this would be too expensive, given that we aim next to project each of them by several techniques, both in 2D and 3D, and compute several quality metrics for each combination.

In additional contrast to the 2D benchmark, we selected datasets which are known, from earlier studies, to exhibit discernible *structure* in terms of clusters of samples. This will be important for our qualitative study (Sec. 6.4) in which we aim to compare how easily such structure is perceived in 2D, respectively 3D, projections. Indeed, selecting some arbitrary dataset that would not have any clear structure would make the qualitative comparison of 2D *vs* 3D projections useless. Secondly, we selected on purpose 7 of the 8 datasets as being relatively low-dimensional (up to 30 dimensions): If 3D projections would not prove better than 2D ones, even for such datasets, then the challenge would be even harder for higher-dimensional ones. The eighth dataset (*Reuters*) was taken as a control sample, to gauge how our results would extrapolate for data having high (intrinsic) dimensionality.

6.3.2 *Projections*

From the 44 projection techniques present in the 2D benchmark, we selected those which could compute, out-of-the-box, both 2D and 3D projections, yielding a total of 29 projection techniques for our evaluation. We excluded techniques which are not open source. Table 5 lists, for these, whether they are (non)linear, accept samples or sample-pair distances as input, project local neighborhoods differently or work globally the same for the entire dataset, their computational complexity, whether they have out-of-sample quality, whether they are deterministic or stochastic, and the public source of their implementation (for replication purposes). Complexity is a function of the number of dimensions n, number of samples N, number of iterations i (for iterative methods), and number of weights w (for deep learning methods). As Table 5 shows, the selected projections cover a wide spectrum of methods.

Table 4: Selected datasets for evaluation and their trait values (Sec. 6.3.1).

Dataset	**Type**	**Samples N**	**Dimensions n**	**Intrinsic dimensionality ρ_n**	**Sparsity γ_n**
Air quality (Vito et al., 2008)	tables	9357	13	5	0.1372
Breast cancer (Wolberg et al., 2021)	tables	569	30	10	0.0059
City pollution (Zhang et al., 2017)	tables	32681	10	6	0.0052
Concrete (Yeh, 1998b)	tables	1030	8	6	0.1773
DefaultCC (Yeh and Lien, 2009),	tables	30000	23	8	0.1070
Software (Meirelles et al., 2010)	tables	6773	12	7	0.0818
Wine (Cortez et al., 2009b)	tables	6497	11	8	0.0023
Reuters (Keras repository, 2021)	text	8432	1000	696	0.9488

Table 5: Selected projection techniques for evaluation and their characteristics (Sec. 6.3.2).

Projection	linearity	Input	Neighborhood	Complexity	Out-of-sample	Deterministic	Implementation
AE (Hinton and Salakhutdinov, 2006)	nonlinear	samples	global	$O(iNw)$	yes	no	Keras
DM (Coifman and Lafon, 2006)	nonlinear	samples	local	$O(N^3)$	no	yes	Tapkee
FA (Jolliffe, 1986)	linear	samples	global	$O(n^3)$	yes	yes	scikit-learn
F-ICA (Hyvarinen, 1999)	linear	samples	global	$O(n^3)$	yes	yes	scikit-learn
G-RP (Dasgupta, 2000)	nonlinear	samples	global	$O(Nn^3)$	yes	no	scikit-learn
H-LLE (Donoho and Grimes, 2003)	nonlinear	samples	local	$O(N^3)$	yes	no	scikit-learn
I-PCA (Ross et al., 2008)	linear	samples	global	$O(n^3)$	yes	no	scikit-learn
ISO (Tenenbaum et al., 2000)	nonlinear	samples	local	$O(N^3)$	yes	yes	scikit-learn
K-PCA-P (Schölkopf et al., 1997)	nonlinear	samples	global	$O(N^3)$	yes	no	scikit-learn
K-PCA-R (Schölkopf et al., 1997)	nonlinear	samples	global	$O(N^3)$	yes	no	scikit-learn
K-PCA-S (Schölkopf et al., 1997)	nonlinear	samples	global	$O(N^3)$	yes	no	scikit-learn
LE (Belkin and Niyogi, 2002)	nonlinear	distances	local	$O(N^3)$	no	no	scikit-learn
LLE (Roweis and Saul, 2000)	nonlinear	samples	local	$O(N^3)$	yes	no	scikit-learn
L-LTSA (Zhang et al., 2007)	linear	samples	local	$O(N^3)$	no	no	Tapkee
L-MDS (De Silva and Tenenbaum, 2004)	nonlinear	distances	global	$O(N^3)$	no	no	Tapkee
LPP (He and Niyogi, 2004)	linear	samples	global	$O(N^3)$	yes	yes	Tapkee
LTSA (Zhang and Zha, 2004)	nonlinear	samples	local	$O(N^3)$	yes	no	scikit-learn
MDS (Torgerson, 1958)	nonlinear	distances	global	$O(N^3)$	no	no	scikit-learn
M-LLE (Zhang and Wang, 2007)	nonlinear	samples	local	$O(N^3)$	yes	no	scikit-learn
N-MDS (Kruskal, 1964)	nonlinear	samples	global	$O(iN^2)$	no	no	scikit-learn
NMF (Lee and Seung, 2001)	linear	samples	global	$O(n^2)$	yes	no	scikit-learn
NPE (He et al., 2005)	nonlinear	samples	local	$O(N^3)$	yes	no	Tapkee
PCA (Jolliffe, 1986)	linear	samples	global	$O(n^3)$	yes	yes	scikit-learn
S-PCA (Zou et al., 2006)	linear	samples	global	$O(N^3)$	yes	yes	scikit-learn
SPE (Agrafiotis, 2003)	nonlinear	samples	global	$O(N^2)$	no	no	Tapkee
S-RP (Dasgupta, 2000)	nonlinear	samples	global	$O(Nn^3)$	yes	no	scikit-learn
T-SNE (van der Maaten and Hinton, 2008)	nonlinear	distances	local	$O(iN^2)$	no	no	Multicore TSNE
T-SVD (Halko et al., 2009)	linear	samples	global	$O(N^2)$	yes	no	scikit-learn
UMAP (McInnes et al., 2018)	nonlinear	distances	local	$O(iN^2)$	yes	no	umap-learn

6.3.3 *Metrics*

Table 6 lists the projection quality metrics we used, which are the most common ones used in the projection literature to gauge the quality of dimensionality reduction (van der Maaten and Postma, 2009; Espadoto et al., 2019). All metrics range in $[0, 1]$ (0=minimal quality, 1=maximal quality). These metrics are explained below.

Trustworthiness M_t: Measures the fraction of close points in D that are also close in $P(D)$ (Venna and Kaski, 2006b), being the inverse of the false neighbors metric in Martins et al. (2014). M_t tells how much one can trust that clusters in a projection represent actual data patterns. In its definition (Tab. 6), $U_i^{(K)}$ is the set of points that are among the K nearest neighbors of point i in $\mathbb{R}^q$ but not among the K nearest neighbors of point i in $\mathbb{R}^n$; and $r(i, j)$ is the rank of the point j in the ordered set of nearest neighbors of i in $\mathbb{R}^q$ (ordering being here given by Euclidean distance). We use $K = 7$, in line with earlier similar work (van der Maaten and Postma, 2009; Martins et al., 2015; Espadoto et al., 2019);

Continuity M_c: Measures the fraction of close points in $P(D)$ that are also close in D (Venna and Kaski, 2006b). It is the inverse of the missing neighbors metric in Martins et al. (2014). In its definition (Tab. 6), $V_i^{(K)}$ is the set of points that are among the K nearest neighbors of point i in $\mathbb{R}^n$ but not among the K nearest neighbors in $\mathbb{R}^q$; and $\hat{r}(i, j)$ is the rank of the $\mathbb{R}^n$ point j in the ordered set of nearest neighbors of i in $\mathbb{R}^n$. As for M_t, we chose $K = 7$.

Shepard diagram correlation M_S: In a scatterplot of the point-pair distances in $P(D)$ *vs* the corresponding distances in D – the Shepard diagram S – points close to a diagonal indicate good distance preservation (Joia et al., 2011). Points below, respectively above, the diagonal indicate distance ranges for which false neighbors, respectively missing neighbors, occur. We measured distance preservation by the Spearman rank correlation M_S of the Shepard diagram. A value of $M_S = 1$ indicates a perfect (positive) distance correlation. Note that, in Tab. 6, the Shepard diagram S is not, formally speaking, a quality metric, since its result is not a scalar-valued function as the other metrics. However, we included its definition in the respective table for ease of reading as S is needed next for computing the goodness metric M_S.

We did not consider additional projection quality metrics in the literature such as metrics which cannot be (easily) aggregated to a single scalar value per scatterplot, *e.g.*, the projection precision score (Schreck et al., 2010), stretching and compression (Aupetit, 2007; Lespinats and Aupetit, 2011), average local error (Martins et al., 2014), and the coranking matrix (Lee and Verleysen, 2009), since we want next to compare *hundreds* of such scatterplots. We also did not consider metrics

Table 6: Projection quality metrics used in our quantitative evaluation (Sec. 6.3.3).

Metric	**Definition**
Trustworthiness (M_t)	$1 - \frac{2}{NK(2n-3K-1)} \sum_{i=1}^{N} \sum_{j \in U_i^{(K)}} (r(i,j) - K)$
Continuity (M_c)	$1 - \frac{2}{NK(2n-3K-1)} \sum_{i=1}^{N} \sum_{j \in V_i^{(K)}} (\hat{r}(i,j) - K)$
Shepard diagram (S)	$\{(\|\mathbf{x}_i - \mathbf{x}_j\|, \|P(\mathbf{x}_i) - P(\mathbf{x}_j)\|)\}, 1 \leq i \leq N, i \neq j$
Shepard goodness (M_S)	Spearman rank correlation of S

which do not make sense for all types of projection, *e.g.*, normalized stress (Joia et al., 2011); and metrics which require labeled data, *e.g.*, neighborhood hit (Paulovich et al., 2008) and the Class Consistency Measure (CCM) (Sips et al., 2009; Tatu et al., 2010).

6.3.4 *Evaluation results*

We evaluated all 29 projection techniques, for their 2D and 3D variants on our 8 datasets using the 3 quality metrics in Sec. 6.3.3. Projection hyperparameters were set to the optimal defaults found in Espadoto et al. (2019). We next analyzed the computed quality metrics from several perspectives.

Figure 6.1 shows the three quality metrics (Sec. 6.3.3) per dataset, projection technique, and 2D *vs* 3D projection variant, sorted ascendingly on trustworthiness per technique, for ease of examination. The metric values for City pollution (DM, SPE, MDS, N-MDS, and LE projections), Air quality (NPE projection), and Defaultcc (DM, SPE, MDS, N-MDS, and LE projections) are missing, as these techniques failed executing on the respective datasets, due to unknown factors (likely, implementation issues of the respective techniques). Overall, from Fig. 6.1, we see a globally small variation across *techniques* – which is fully in line with the results of Espadoto et al. (2019). More interestingly, the 3D techniques scored almost always better *but only marginally* compared to their 2D counterparts. All these findings do not seem to depend on the *dataset*. These observations strongly suggest that 3D projections consistently bring some, but marginal, increase of quality *vs* their 2D counterparts, regardless of the technique, dataset, and metric being used.

Figure 6.2 refines these insights. Image (a) shows the averages trustworthiness, continuity, and Shepard correlation, for each 2D projection technique (circles), respectively 3D technique (triangles). Continuity is slightly higher for 3D techniques – on average, 0.02 over all projection techniques. Trustworthiness shows the same trend – 3D techniques are 0.05 more trustworthy than 2D ones on average. While Shepard correlation varies more per technique, 3D projections still score slightly better than 2D ones, 0.03 more on average. Image (b) merges the trustworthi-

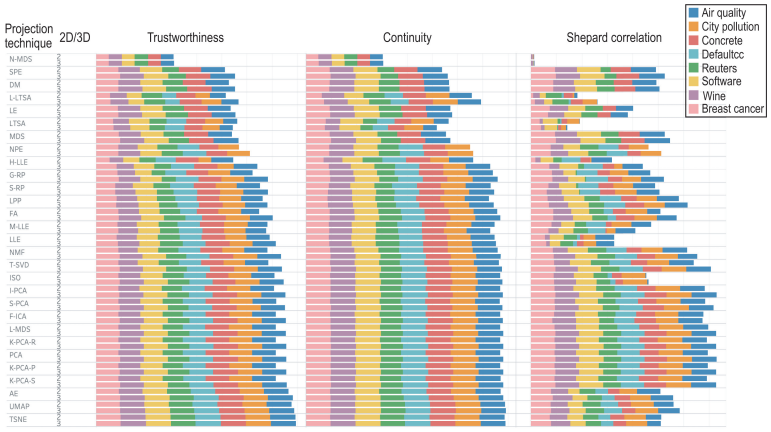


Figure 6.1: Quality metrics per projection technique (rows), dataset (colors), and projection dimension (2D *vs* 3D, second column), sorted ascendingly on technique trustworthiness.

ness and continuity plots in image (a) showing a positive correlation of the two metrics over all projections. We placed the origin of this plot at 0.5×0.5, since none of the two metrics is below this value. Globally, we see that N-MDS scores poorest, followed by LTSA. The best scoring techniques are t-SNE, UMAP, and AE. For these, however, the quality gain given by 3D projections is negligible. The technique showing the largest gain between 2D and 3D is H-LLE, where 3D adds about 12% in trustworthiness and 8% in continuity, respectively. The stacked bars for H-LLE in Fig. 6.1 show us that this gain is independent of the dataset. Hence, for H-LLE, the use of a third dimension brings some significant added value.

Summarizing the above, we see that the use of a third dimension brings only minimal increase of quality metrics for all projections being studied, over all studied datasets, except H-LLE, whose 3D variant scores about 10% higher quality than its 2D variant.

6.4 QUALITATIVE STUDY

The analysis in Sec. 6.3 showed that 3D projections do not come with significant higher quality metrics than their 2D counterparts. However, we cannot say, based solely on this, that they do not have added value. Indeed, the quality metrics used in Sec. 6.3 capture only a fraction of the expressive nature of a projection. Many other quality metrics exist, for example those used to capture the visual separation of clusters in projections (Albuquerque et al., 2011; Sedlmair et al., 2013; Motta et al., 2015). For labeled data, the so-called Class Consistency Measure (CCM) (Sips et al., 2009; Tatu et al., 2010) was shown to model well the way hu-

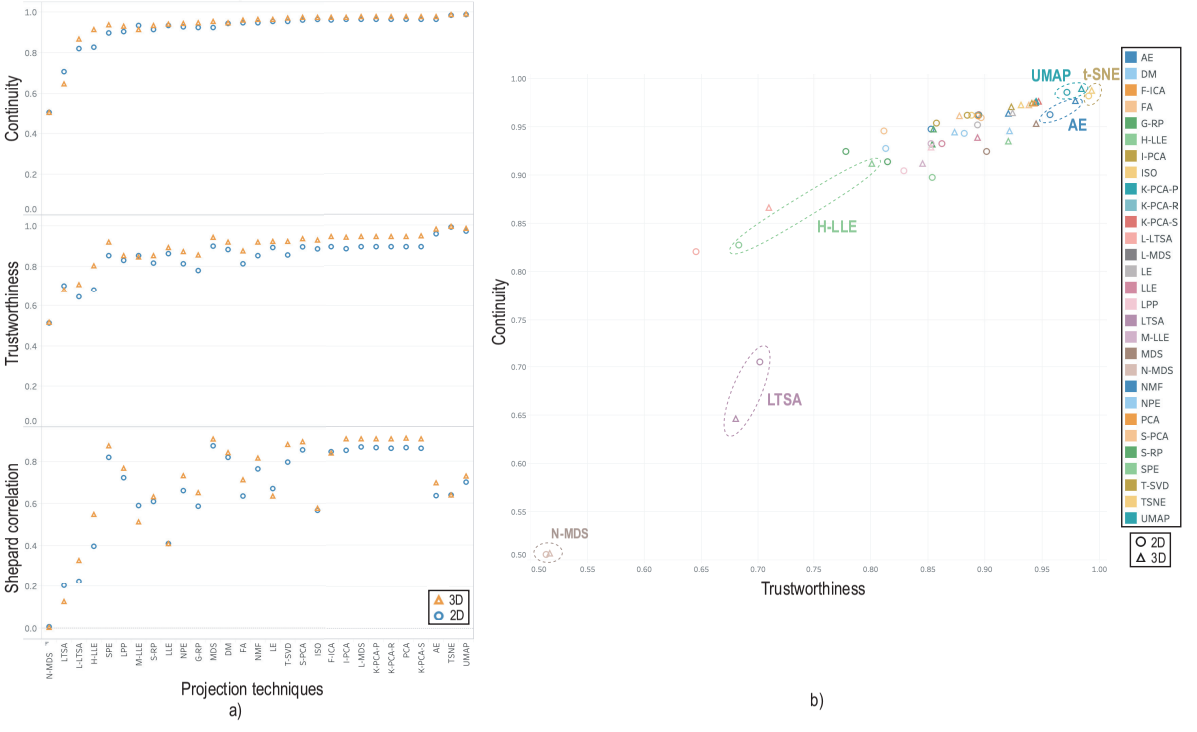


Figure 6.2: a) Continuity, trustworthiness, and Shepard correlation, averaged for all datasets, for 2D and 3D projections. b) Correlation of trustworthiness *vs* continuity, averaged per projection over all datasets, for 2D and 3D projections.

mans visually separate same-label clusters in a projection (Sedlmair and Aupetit, 2015). However, computing such cluster-separation metrics assumes one to project *labeled* data and also that the respective data contains well-separated same-label point groups. This is not always the case for datasets which are explored using projections, as also noted in Espadoto et al. (2019). Moreover, the actual way users would perceive the added value (or lack thereof) of 3D projections cannot be fully captured by metrics such as the ones mentioned above.

We further gathered insight in how 2D and 3D projections differ, by a three-part qualitative study, in a bottom-up fashion – starting from an easy task and proceeding with more complex ones – as follows.

6.4.1 *Identifying visual structure*

We first considered the task of using the projection to find any apparent data structure depicted therein. For this, we looked at whether the projection is separated into distinct *clusters*, since this is one of the main use-cases behind visual exploration of projections (Nonato and Aupetit, 2018; Sedlmair et al., 2013; Poco et al., 2011). Note that we did not consider labels in this task, but rather only whether the projection captures the 'modes' of the underlying data distribution. More precisely, we aimed to see whether 3D projections reveal better such existing separation – if present in the data – than their 2D counterparts. For this, we created scatterplots of all the 8 datasets in Tab. 4 projected in 2D and 3D by all the 29 projection techniques in Tab. 5. Next, we visually compared the corresponding 2D and 3D projection plots – to be more exact, the 2D plots with 2D views (from selected viewpoints) of the 3D plots. In all plots, we colored points based on the ID of the corresponding high-dimensional points using a heat colormap. This allowed us to see whether different plots place points close to each other in similar ways – if so, they will exhibit similar color gradients. Note that this should not be confused with the typical color-by-attribute-value mode used in exploring projections, whose aim is different, *i.e.*, to explain patterns in a projection by data *values*. Next, we interactively rotated the 3D plots aiming to find the view which best conveys separated clusters. Finally, we aligned this view (by means of manual rotation around the view axis and viewport scaling) to best match the corresponding 2D projection, for visual comparison purposes.

Figure 6.3 shows the results of this evaluation for the *Wine* dataset, with 2D projections always to the left of their 3D counterparts for the same technique. Results for H-LLE, LTSA, and M-LLE are omitted since these projections create a very large amount of point overlap, making their visual exploration useless (both in 2D and 3D). Similar results to Fig. 6.3 for all studied datasets are in the supplementary material, including videos showing the 3D projections from multiple viewpoints.

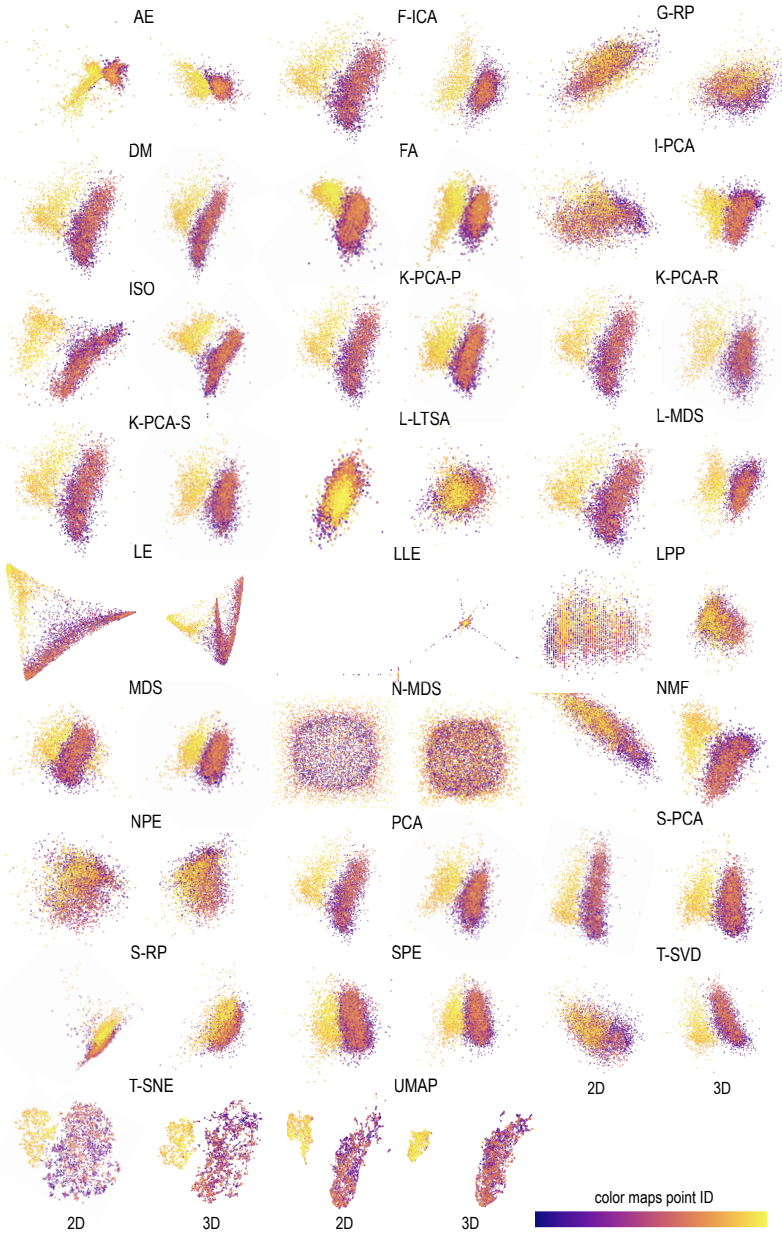


Figure 6.3: Visual comparison of 2D *vs* optimal views of the 3D projections for the task of identifying separated data clusters, *Wine* dataset (Sec. 6.4).

These images convey us several interesting insights, as follows.

Data patterns: The vast majority of projections show that the *Wine* dataset is roughly split into two clusters (red-purple, respectively yellow points in Fig. 6.3). This is in line with other works that studied this dataset (Coimbra et al., 2016; da Silva et al., 2015; van Driel et al., 2020). As a baseline, this tells us that our study is properly set up to next explore the other projections.

2D *vs* 3D projections: In almost all the cases, the 3D projections show the same patterns as their 2D counterparts. The exceptions are I-PCA, NMF, and (partly) T-SVD. For these techniques, the 2D plots do not show any data structure, whereas the 3D plots show a clear separation of the two underlying data clusters. Separately, we see that 'good' projection techniques work equally well in 2D and 3D to create visual structure – or equally poorly. For the latter case, we have N-MDS, L-LTSA, LLE, LPP, NPE, and S-RP. These techniques are not able to identify any visually salient patterns in the data, neither in the 2D nor in the 3D case.

Projection quality: As explained in the beginning of this section, quality metrics are not to be used as a sole mean to assess whether a projection is useful in conveying data patterns. Figure 6.3 confirms that: We see a large variation in the ability of projections to find data patterns, ranging from very strong cluster separation (T-SNE and UMAP) to almost no structure (N-MDS, NPE). This is only partly reflected by the metric values (Fig. 6.1): While the techniques that score poorly in finding visual structure (N-MDS, L-LTSA, LLE, LPP, NPE, and S-RP) also have some of the lowest quality metrics, AE scores third-highest metric-wise, but arguably shows a poorer visual separation of data structures than MDS which has the 7^{th} lowest metric values.

Summarizing the above, we found that 3D projections produce roughly the same visual patterns as their 2D counterparts, these patterns depending far more on the projection technique being used than on the dimensionality of the output scatterplot (2D or 3D). Also, producing the same informative views cost more time for the 3D projections, since a suitable viewpoint must be found by interactive rotation, whereas the 2D projections required no user interaction.

6.4.2 *Explaining visual structure*

Our first evaluation (Sec. 6.4.1) showed that 3D projections seem, overall, to be able to generate similar amounts of *visual* structure to their 2D counterparts. However, by itself, this does not directly tell us that 3D and 2D projections are equally effective in understanding *data* structure. Indeed, visual structures in a projection need explanations to be

further understood and interpreted by users (Sec. 6.2.4). Without these, a 'raw' projection, even when showing some visual structure, is of little use. We next studied how the variance-based explanation of projections of da Silva et al. (2015); van Driel et al. (2020) augments the added-value of 3D *vs* 2D projections. This explanation colors projected points $P(\mathbf{x}_i)$ by the identity of the dimension $\mathbf{x}^j$ which has the least variance over a small neighborhood around $P(\mathbf{x}_i)$. Color brightness encodes the explanation confidence, *i.e.*, how much of the total variance (over all n dimensions) in a neighborhood in $P(D)$ is explained by the color-coded dimension there. Among other projection explanations (Sec. 6.2.4), we selected this one since it works generically for any projection technique, acts locally per projection neighborhood (so, can handle both local and global projection techniques), is fast and simple to compute, and is easy to introduce to users. We implemented this explanation for 3D projections by extending the earlier work (da Silva et al., 2015) that considered 2D projections only. We next applied the explanation to all our 2D and 3D projections computed as outlined in Sec. 6.3.

Figure 6.4 shows a selected subset of 2D and 3D projections for the *Wine* dataset (for space reasons, all results are in the supplementary material, see reference (Tian et al., 2021d) for details) color-coded by the Da Silva explanation. Points are rendered with blended splats, following da Silva et al. (2015). Legends indicate the data dimensions color-coded in the explanations. Since we wanted to test how the Da Silva explanation helps understanding visual structure, we separated projections in those found (Sec. 6.4.1) to exhibit a clear visual structure in 2D (Fig. 6.4 top half), respectively those which showed such structure far less clearly (Fig. 6.4 bottom half).

A first analysis of Fig. 6.4 shows that, for the top projections, the patterns visible in the 3D projections are quite similar to those shown by their 2D counterparts. For instance, UMAP (2D) separates the data into two clusters. The color-based explanation further splits the larger left cluster into wines that are similar mainly because of chlorides (pink), respectively alcohol (red). The smaller right cluster is nearly completely explained as wines having similar sugar content, apart from a few points at the bottom which are wines having similar alcohol percentages. The 3D projection created by UMAP tells us essentially the same story. The same situation occurs for FA, where the 2D and 3D projections are both split into essentially three zones explained by alcohol (pink), sugar (yellow), and chlorides (red). This suggests that 3D projections do not help gaining more, or different, insights as compared to their 2D counterparts.

However, comparing the 2D and 3D projections in Fig. 6.4 has a problem: Different colormaps are used to encode the same dimensions for the same dataset. For example, the 2D Isomap projection of the *Wine* dataset in Fig. 6.4 (top left) uses pink, yellow, red, and green to encode alcohol, chlorides, sugar, and volatile acidity, respectively. The 3D Isomap

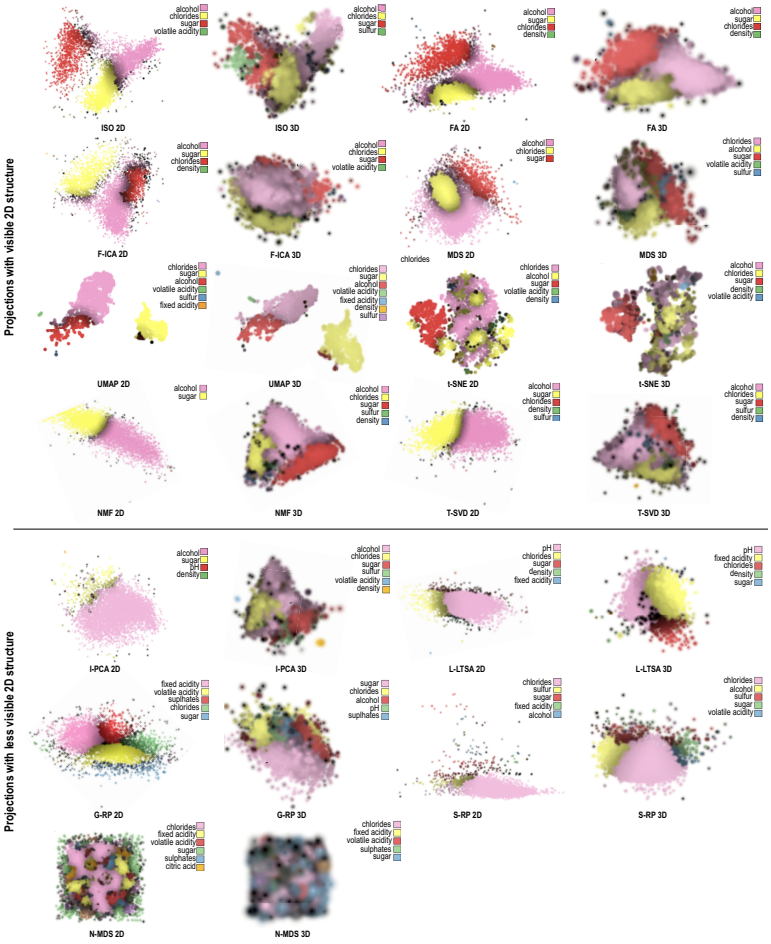


Figure 6.4: Comparison of 2D and 3D projections explained by dimension variance, *Wine* dataset. Dimension-to-color mapping is computed per individual projection (Sec. 6.4.2).

projection of the same dataset uses pink, yellow, and green for the same dimensions, but allocates green to sulfur. This is inherent to how the algorithm in da Silva et al. (2015) works: Each projected point is assigned a dimension that best explains the neighborhood around it; next, for each dimension $1 \leq j \leq n$ of the dataset, the number of projected points e_j that choose dimension j as best is computed. Finally, the values e_j are sorted descendingly and the first C dimensions that emerge from this sort are mapped to a categorical colormap of $C = 8$ colors. This way, colors are allocated to those dimensions which can explain the most projected points. Since 2D and 3D projections (of the same dataset) have different structures, their top-voted C dimensions can differ, leading to the same dimension being mapped to different colors and/or the same color allocated to different dimensions.

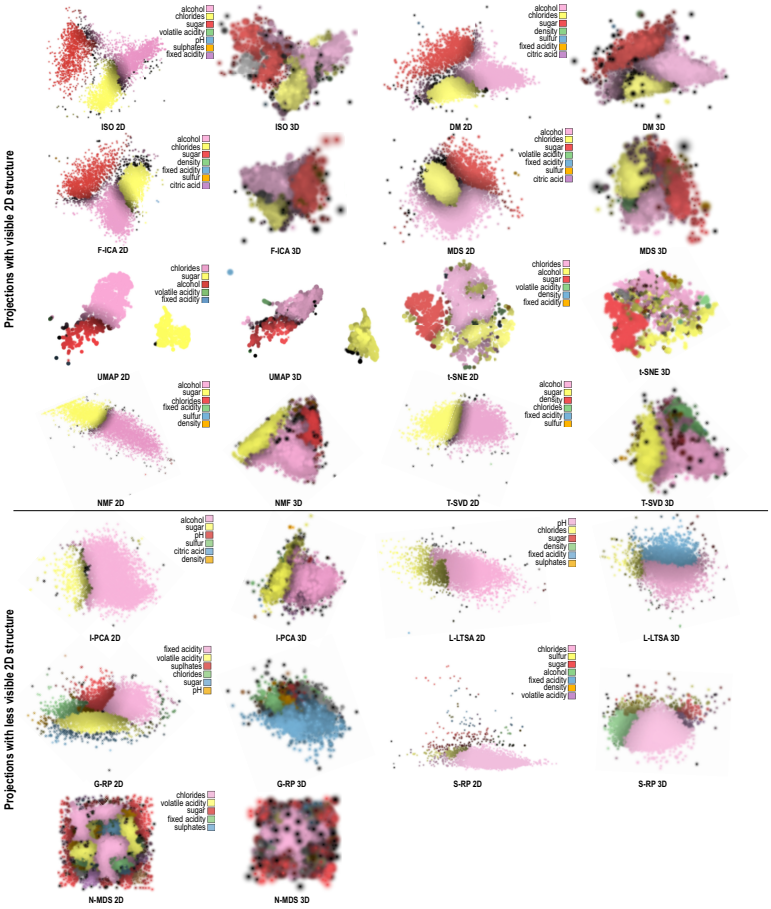


Figure 6.5: Comparison of 2D and 3D projections, *Wine* dataset. Same as Fig. 6.4, but using the same dimension-to-color mapping for 2D and 3D projections created by the same technique.

To remove this confusion, we redid in Figure 6.5 the plots in Fig. 6.4 using the same dimension-to-color mapping for each *pair* of 2D and 3D projections created by the same technique. For this, we ran the Da Silva algorithm once, *e.g.*, when explaining the 2D projection, and saved the dimension-to-color mapping it produces. Then, we ran the algorithm for the 3D projection. If this run selected dimensions already assigned to colors in the first run, then we used the colors assigned the first time; if new dimensions are mapped to colors (by the second run), then we allocated colors not used by the first run.

Looking at Fig. 6.5, the difference between the top projections (found earlier to exhibit visible structure in 2D) and the bottom ones (found earlier to have less visible 2D structure) becomes now clearer: For the top projections, we see nearly the same explanations for the 2D and 3D variants of the same technique; there is little added value apparent in using a 3D projection instead of a 2D one, the structures shown by the 3D variant were already visible in the 2D variant. For the bottom projections, the situation is slightly more nuanced. 2D and 3D projections often show the same main explanation patterns, see *e.g.* the yellow (left) and pink (right) clusters present in both the 2D and 3D I-PCA variants (Fig. 6.5, bottom). The 3D projections often introduce additional explanations which were not easily visible in the 2D variants, see *e.g.* the blue fixed acidity cluster for L-LTSA (3D) or the green and red clusters for alcohol and sugar respectively for S-RP (3D). In the extreme case of N-MDS, which had an extremely poor explanation in 2D, using a 3D projection does not improve the situation at all. To conclude, this analysis tells that 3D projections, even when explained (by the Da Silva method), do not bring significant extra value as compared to their 2D counterparts.

6.4.3 *Expert evaluation*

To gain more insights in how explained 3D projections compare to their 2D counterparts, we performed a user evaluation, detailed next.

Participants: We asked four data scientists to take part in our study. All were familiar with dimensionality reduction, and with the Da Silva technique, and worked in information visualization for 2, 3, 9, and 13 years, respectively. They were instructed first in how to use a visualization tool that allows examining the 2D or 3D projections via zoom, pan, rotation, and brushing points to see their attributes. They were also offered videos showing the respective projections visualized in the tool, for convenience. We precomputed all projections ourselves so that all users would see the same results and would not be bothered with tweaking projection-algorithm parameters.

Data: We computed 2D and 3D projections for the first 7 datasets in Tab. 4 using all 29 techniques in Tab. 5. We did not use the *Reuters* dataset since this is very high-dimensional (1000 dimensions) and thus not suitable for the Da Silva explanatory technique. Also, 11 dataset-technique combinations failed to compute (see Sec. 6.3.4). Hence, a total of $7 \times 29 - 11 = 192$ projection-pairs were offered for investigation to the users.

Tasks: As outlined earlier, the main use-case behind the Da Silva explanatory technique and its variants is to allow users to visually 'partition' a projection into different zones, each being explained by a different dimension. Note that such zones need *not* be separated by whitespace, *i.e.*, they can be different, and usually are a superset of, the visual clusters that projections are typically used to find. For example, the UMAP (2D or 3D) projections in Fig. 6.5 show, each, two visual clusters (red-pink and yellow), but three zones (red, pink, and yellow). Given this use-case, we next asked the users to study the provided 2D-3D projection pairs by comparing them side by side, and to note down how they would rank the variants, using four classes:

1. the 2D and 3D variants are **equally good** and informative;
2. the **2D variant** is clearly preferred;
3. the **3D variant** is clearly preferred;
4. both variants are **equally poor** (hard to understand, thus useless).

For classes 2 and 3 above, we also asked the users to note down why they preferred one variant against the other and save screenshots of the respective variants. We also asked the users to write down, at the end of the study, any global comments they had concerning the use of 2D *vs* 3D explained projections. There was no hard time limit imposed for the study – the users could stop when they wanted.

Results: From the projection-pairs offered to study, 43 were marked in class 4, *i.e.*, hard to understand and further useless. From the remaining ones, about 80% were marked in class 1 (2D and 3D variants address the task equally well). The remaining 20% was roughly evenly split into class 2 (2D variant clearly preferred) and class 3 (3D variant clearly preferred). We did not find correlations between these classes and the projection methods and/or the datasets. We found, however, more interesting facts when reading the comments given by the users to their rankings. We list the most salient findings next – see also Fig. 6.6 for user-made screenshots supporting these findings.

Perceived advantages of 3D projections

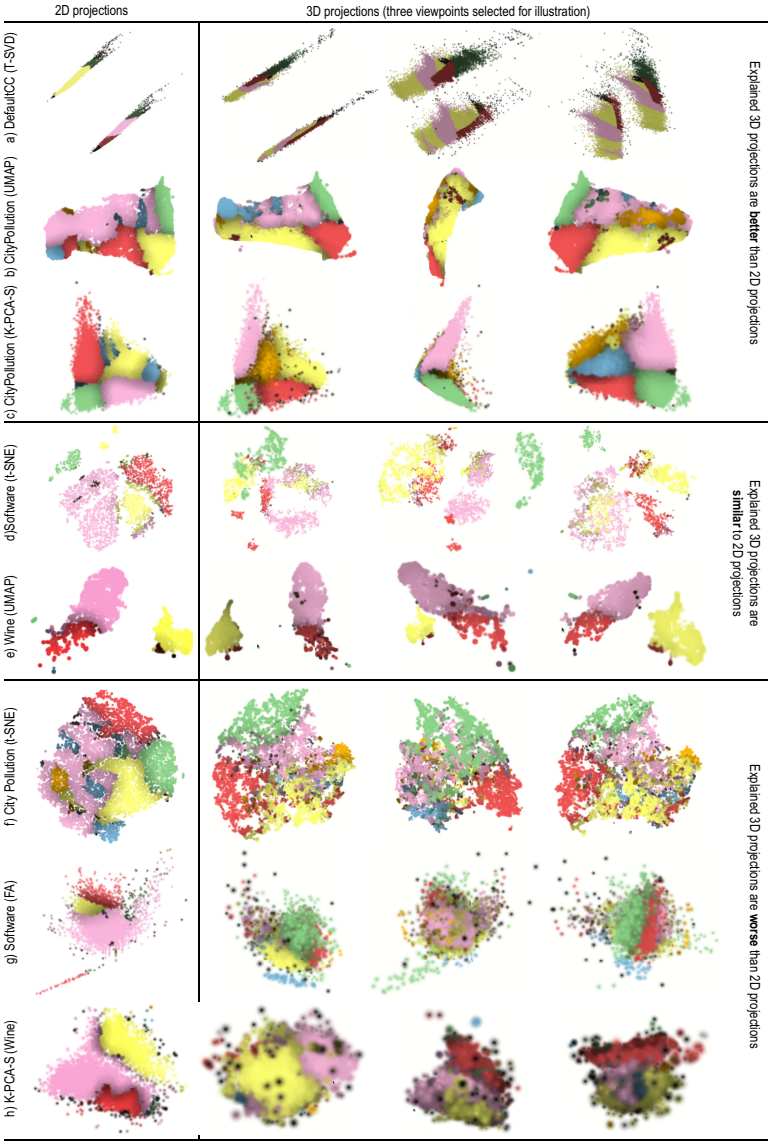


Figure 6.6: Selected examples from the user evaluation showing how variance-explained 3D projections can be better than, similar to, or worse than, 2D projections (Sec. 6.4.3).

- 3D projections spread the points over a larger space, so can show more complex patterns. Figure 6.6a shows an example: the 2D T-SVD projection essentially creates two narrow bands along which little structure is visible. The 3D variant creates two *plane*-like structures that can show more explanation details. The 3D dimension also increases the chance that more variables will be involved in the explanation, which is good, since the explanation becomes more fine-grained. Figures 6.6b,c show this for *CityPollution* projected with UMAP and K-PCA-S: In both cases, the 2D projection cannot really show the orange cluster (points similar due to the SO_2 dimension). This is because points are too tightly packed in 2D, so there is no room to 'spread out' this dimension. In 3D, the projections yield a similar (triangular-shape) surface to the 2D case. Yet, the additional spatial dimension allows spreading out points above the surface, so the orange cluster becomes visible. Also, the third dimension gives more chance for visual cluster separation as compared to 2D projections.

- 3D projections were found to give the user a sense of control in terms of selecting which are interesting views. While no ideal viewpoint can be found in general, different viewpoints could be used to show different parts of the data in turn, one by one. This allows further finding and exploring structures (one by one) which would otherwise be occluded, and have no chance to show up, in a 2D projection – see *e.g.* the three viewpoints for Figs. 6.6b,c; only in two of these is the orange cluster visible. Overall, 3D projections were found more versatile than 2D ones, being able to tell different stories about the data, depending on the chosen viewpoint.

Perceived advantages of 2D projections

- One user remarked that the key advantage of 2D projections was their ease of use. No interaction is required to examine them, while one can get lost or frustrated in the process of zooming, panning, and viewpoint rotation for 3D projections. As such, this user noted that, in about 80% of the class-1 cases (2D found similar to 3D), this did not take into account the interaction effort. If this effort were to be considered, then those cases should be marked as class 2 (the 2D variant is preferred). Quoting from this user: "Both 2D and 3D are fine. Yet, I prefer 2D because it gives very clear results without further interaction needed." Figures 6.6d,e show two such cases. The visible clusters and their explanations are very similar in 2D and 3D, so, for these cases, the 3D variant does not add any perceived value.

- Some projection techniques, in particular t-SNE, were consistently found to create clearer explanations in 2D than in 3D –

something already visible in Figs. 6.4 and 6.5. This is an important observation, since t-SNE is known as a very high-quality projection. Such quality would, thus, be lost if using the 3D variant. Figure 6.6f shows this. The 3D t-SNE projection actually spreads points on a ball-like surface, with some points also being placed inside. It is very hard, even with interaction, to find out which points are close together on the same 'side' of the surface.

- 2D projections were definitely preferred in the cases where the nature of the data would create densely-packed clusters. These would map to close groups of points in the 2D projection (which are fine). In 3D, however, this would create a densely packed 'hairball' of regions explained by the different variables (Fig. 6.6h). Occlusion would then prevent the user from discovering interesting structures and/or explanations inside such a 3D structure.
- Outliers were also found easier to spot with 2D projections. They would appear as points separated by large amounts of whitespace from the high-density 'core' of the projection. In 3D, however, outliers could appear *in front* or *behind* the high-density core, and thus be hard to spot (Fig. 6.6h).

6.5 DISCUSSION

We discuss several points concerning our findings and methodology, as follows.

Quantitative results: The comparison of 2D *vs* 3D projection quality metrics discussed in Sec. 6.3 is, to our knowledge, the first study of its kind in projection literature. Overall, our results show that t-SNE, UMAP, AE reach the best metric values for 3D projections, similar to the results found earlier for projections (Espadoto et al., 2019). Our main novelty is to show that, metric-wise, 3D projections are only *marginally* better than their 2D counterparts – a fact which, to our knowledge, was never quantified by quality metrics.

Pattern identification: Our first qualitative study (Sec. 6.4.1) showed that 3D projections do not bring significant added value over their 2D counterparts in terms of finding data structures. 3D projections either show the same structure type, or otherwise do not show any structure at all, similar to their 2D counterparts. Our findings match those of Sedlmair et al. (2013) – but generalize them, since we explored 29, as opposed to just 4, projection techniques (PCA, Robust PCA, MDS, and t-SNE) used by Sedlmair et al. (2013); also, we used optimal parameter presets for the studied techniques, something not considered by Sedlmair et al. (2013). Our subsequent qualitative study (Secs. 6.4.2, 6.4.3) showed that, when augmented with the Da Silva explanation,

3D projections can, in some cases, show more insights in the data than their 2D counterparts, *e.g.*, they partition the dataset into more zones explained by more data dimensions. However, in most cases, the patterns shown by 3D projections are very similar to the 2D ones; and 3D projections introduce additional challenges such as occlusion and additional user effort for exploration.

Choice of projection techniques: An important point must be made concerning the choice of studied projection techniques and the presented findings. Clearly, not all techniques are equally good for projecting any dataset. Espadoto et al. (2019) have extensively documented this, by benchmarking 44 such techniques against 19 datasets for 2D projections. Their results showed only small variations of the projection quality, measured by 7 quality metrics. As such, the question of why certain projection techniques are better than others cannot be gauged simply by quality metrics, as already argued in Sec. 6.4. A separate question is how projection techniques perform with respect to other dataset traits, beyond intrinsic dimensionality and sparsity (see Sec. 6.3.1). For instance, the distribution of samples in a dataset can be an important trait that characterizes the quality of a projection technique. We do not examine this aspect in this chapter for the following reasons:

- The question "which projection technique is the best for a given dataset type" is not in our scope. Rather, as explained in Sec. 6.1 and next in this chapter, our research question is how can visual explanations and/or 3D projections bring added value. These questions do not focus on comparing projection *techniques* against each other, but the same techniques against their instances with or without visual explanations, and with or without a third dimension;

- Comparing 'raw' projection techniques against each other has been done in detail by Espadoto et al. (2019). As said earlier, we aim here not to compare raw techniques, but techniques with (or without) the additions of a third dimension and/or visual explanations;

- It is inherently hard to link the performance of projection techniques to the 'nature' of a given dataset. We did this by using the so-called dataset traits (dimensionality, intrinsic dimensionality, and sparsity) outlined in Sec. 6.3.1. Of course, additional traits can be defined, such as the nature of the distribution that characterizes the samples in a dataset. However, doing this is far from trivial: There are, to our knowledge, no established 'classes' of cannonical distributions for nD datasets. The goal of characterizing how projection techniques cope with various such distributions is definitely an interesting topic to study, but one out of scope of

our chapter which focuses on comparing 2D *vs* 3D projections, with *vs* without visual explanations.

Availability: All our experimental results, including snapshots of the 2D projections, videos of exploring the 3D projections, are available online (Tian et al., 2021a). The source code of the visualization tool that implements the variance-based projection explanations, written in Rust using OpenGL, is publicly available at (Tian et al., 2021b).

Qualitative aspects: The question whether 3D projections are preferable to 2D projections involves multiple aspects. We can classify these into objective and subjective ones. Objective aspects include how the two projection types fare against formal quality metrics. In this respect, we showed that 3D projections provide only marginal advantage with respect to their 2D counterparts (Sec. 6.3). Subjective aspects involve, among others, personal preferences of users, as detailed in Sec. 6.4.3. As the question asked to our expert users was quite broad – namely, which projection (2D or 3D) they find the most informative – there was more freedom for users to react by listing a wide range of reasons for their preferences. Still, even with this freedom, the results showed that, from the 192 projection-pairs offered to study, only 15 were marked as 3D clearly preferred to 2D, and 15 were marked as 2D clearly preferred to 3D. This suggests that the impact of personal preference in the choice of 2D *vs* 3D projections is quite small. It is, still, interesting to explore the reasons underlying personal preferences, such as the 'sense of control' and ability of creating different stories about the data (Sec. 6.4.3). Better understanding these aspects can lead to the creation of more engaging exploration tools, *both* for 2D and 3D projections.

Limitations: As any evaluation work in visualization, ours has several limitations. We only explored 8 (real-world) datasets, and considered only relatively simple tasks such as cluster separation identification. However, we argue that, if even for such simple datasets and tasks 3D projections cannot show a clear added-value *vs* their 2D counterparts, then this becomes even harder for more complex situations. We believe that refining our findings with more specific (types of) datasets and tasks is a promising direction for future work, which would either highlight use-cases where 3D projections are really superior to 2D ones, or conclude even more firmly that the addition of a third dimension does not bring added value.

A more important limitation regards our expert evaluation (Sec. 6.4.3), which involved only four experts and a general task of ranking projections in terms of being more or less informative. It can be certainly argued that defining more precise tasks, *e.g.*, finding a specific subset of data points which are similar due to a given condition on the data attributes, and measuring the task accuracy and completion time, is needed to refine our insights. However, we also

argue that our preliminary evaluation presented here is valuable in a *formative* sense. Our evaluation is a *qualitative* assessment and, as such, has no generalizable power – we are not aiming to 'prove' or use inferential statistics to claim something. Rather, the purpose is to derive preliminary observations that will inspire future work to generate (and thereafter test) hypotheses. Also, for basic usability studies, it is common to have a limited number of participants (typically 2 to 5). In our case, we further focused on *expert* evaluators; such studies are also common to have a small sample, both due to feasibility and due to emphasis on in-depth analysis. Our formative study allowed us to discover several specific cases where certain 3D projection techniques produce more visual structures of interest than their 2D counterparts (see Fig. 6.6). Moreover, this formative evaluation allowed us to discover that the preference for 2D *vs* 3D projections involves a wide range of factors, going beyond what quality metrics can capture. For example, 3D projections are listed as superior to 2D ones as they invite the user to explore the data more by interactively choosing various viewpoints, something that 2D projections cannot do. We aim to further refine these insights by a formal evaluation which involves techniques and tasks that can exploit the perceived advantage of 3D projections.

6.6 CONCLUSIONS

We presented a multi-faceted comparison of 2D and 3D dimensionality-reduction methods, or projections, for the purpose of finding patterns in high-dimensional data, with the aim of finding added-value (or the lack thereof) for using the third dimension in the scatterplots used to explore such data. As a benchmark, we used 29 projection algorithms and 8 datasets. Our first facet – a quantitative study of three quality metrics – showed consistent, but marginal, added value of the 3D projections. Our second facet – a study in finding visual patterns depicted in the projection – showed that 2D and 3D projections fare almost identically. Our third facet added visual explanations (in terms of attribute variance) to the compared 2D and 3D projections, and showed that both have roughly the same ability in showing very similar patterns. Finally, we executed a user evaluation to elicit additional findings on how 2D and 3D projections compare. We found that, overall, both projection types are found equally insightful, but the 3D ones generate additional challenges and effort.

Summarizing the above, there is little consistent evidence that 3D projections would structurally add value to high-dimensional data exploration atop what 2D projections can do. Still, our study also highlighted several cases where the third dimension does make a difference – in showing more visual structure, more detailed explanations, or engaging users in the data exploration. We aim to refine these findings in several directions. First, we want to test more explanatory tools on

both 2D and 3D projections to see whether some of them can further leverage the third dimension. Secondly, we want to refine the analysis of the cases where 3D projections were found to be better than 2D ones, and thereby develop specialized projection-and-exploration methods that can bring extra value atop what standard 2D projections can deliver. Finally, and in support of both these future work directions, we aim to design more fine-grained controlled experiments where more users than in the current study are given specific quantifiable tasks to execute using 2D and 3D projections in order to compare more precisely their advantages and limitations.

7

CONCLUSIONS

We now conclude our work on our research described in this thesis. For this, we revisit our main research questions and also the way we attempted to answer these.

Recalling the points made in Chapter 1, our key research question was how to improve the exploration and understanding of 2D and 3D visualizations, of low- or high-dimensional data. The unifying element of the work in this thesis is, thus, the dimensionality of the *representation* of the visualized data, which is, namely, 3D *vs* 2D. However, we found out, following the hypotheses outlined in Chapters 1 and 2, that the dimensionality of the *actual* data makes a significant difference on how we can assist users in exploring and understanding it.

We support the above statement by the following findings. We first examined the exploration of low-dimensional, spatial, data – more precisely, 3D spatial datasets such as meshes and point clouds whose coordinates map directly to data dimensions. We found out that, for such datasets, a key challenge is finding good viewpoints to examine them from, and to address this challenge, a major missing component are instruments to flexibly specify rotations around arbitrary axes in 3D. To address this, and thus RQ1, we presented, in Chapter 3, a mechanism for specifying 3D rotations around a wide variety of axes defined in the visualization space. We efficiently and effectively computed such local rotation axes by leveraging the summarization power of 2D binary-image medial descriptors or skeletons. We extended these 2D skeletons with depth information to provide approximations of the above-mentioned 3D local rotation axes. Our proposed technique allows one to specify such complex 3D rotations, we argued, far more easily than existing techniques, by simple point, click, and drag gestures. We compared our novel rotation method to the classical virtual trackball rotation mechanism, both in isolation and in combination, in a controlled user study in Chapter 4. We measured the added value of our proposed rotation technique by a formative study (to elicit main concerns from users) followed by a controlled user study. Results showed that, when combined with the classical virtual trackball rotation, our method leads to better results (in terms of task completion times) and higher user satisfaction than trackball rotation alone. Also, our method is easy to learn and does not carry a significant learning or execution cost for the users, thereby not increasing the costs of using standard trackball rotation.

Concluding this first part of our work, and also RQ1, we state that improvements are certainly possible to existing 3D viewpoint manipulations tools. While our skeleton based rotation showed some clear

advantages, it also appeared not to be able to fully replace virtual trackball rotation. More specifically, we found that skeleton-based rotation is slower than trackball rotation when used alone (for completing a given task); however, when both rotation mechanisms are offered to the user, and the user can freely switch between the two during the given task, the addition of the skeleton-based rotation results in a decrease of the time needed to accomplish that task. Our work, as such, justifies the statement that current 3D viewpoint manipulation tools are still limited; and provides evidence that novel tools, such as our skeleton approach, can provide added value. At a high level, we believe that the main added value of our work showed the unexplored potential of novel interaction mechanisms for 3D viewpoint manipulation and, in particular, the potential value of mechanisms that use the inherent local structure of the visualized 3D data.

The second part of our work focused on the visual exploration, also by 3D visualizations, of high-dimensional data using dimensionality reduction (DR) methods. This covers our research question RQ2. Concerning this, we focused on the task of explaining the meaning of visual structures present in such projections by means of the underlying data dimensions. To do this, and thus answer RQ2, we first addressed the simpler case of 2D projections in Chapter 5. Such 2D projections do not have the additional challenge of choosing a suitable viewpoint for examination. We address the explanation of these projections by developing several so-called local explanation techniques which label neighbor points in the projection by their shared data-related characteristics. The resulting visualizations effectively split the projection point cloud into a number of differently colored and shaded zones, where each zone can be effectively explained in terms of the original data dimensions. We show how our additional explanation mechanisms complement existing projection explanation mechanisms and lead to a better understanding of data represented by such 2D projections. More importantly, we argue that there is no *single* projection explanation mechanism which can fully expose to the user what the meaning of such local structures in the projection is. Rather, it is the examination and combination of several such mechanisms, including the ones we proposed, that contributes to forming a good picture of the data for the user. At a higher level, we thus argue that understanding DR projections cannot be done by any 'single' technique; several techniques, each addressing a different aspect of the data, need to be used in turn by the explorer to form a holistic picture. This is, fundamentally, not surprising – after all, many (tens or hundreds) of data dimensions are 'collapsed' into such a projection. Hence, something is needed to get their meaning back in the projection – such as our proposed explanatory techniques.

We next moved to address the use of 3D DR projections in Chapter 6. More precisely, given our earlier work, we investigated how well 3D projections fare against the use of 2D projections for a number of datasets

and projection techniques. We first approached this comparison in a classical way, using projection quality metrics known from the literature. This quantitative comparison showed consistent, but marginal, added value of the 3D projections. We next moved to comparing 2D *vs* 3D projections in terms of the visual patterns one can detect in such projections. This study showed that 2D and 3D projections fare almost identically. While this may not appear as surprising, we are not aware of a similar controlled experiment that compared 2D *vs* 3D projections.

Next, we compared 2D *vs* 3D projections when both are leveraged by the explanatory techniques we introduced in Chapter 5, since, as argued there, one can best understand what a projection means only when it is suitably explained. Our third study added visual explanations (in terms of attribute variance) to the compared 2D and 3D projections, and showed that both have roughly the same ability in showing similar patterns. Finally, we executed a user evaluation to elicit additional findings on how 2D and 3D projections compare. We found that, overall, both projection types are found equally insightful; and that the 3D projections generate additional challenges and effort but also can, in some cases, expose more patterns than the 2D projections can. Also, our findings showed that users can get more *engaged* in exploring 3D projections than the 2D ones. This is a significant signal that shows that, when appropriately supported by exploration mechanisms, 3D projections can have an end-to-end added value compared to their 2D counterparts.

Concluding this second part of our work, and also our answer to RQ2, we can state that explanation techniques (for both 2D and 3D projections) are definitely a *valuable* instrument in the toolkit of data scientists using projections. Without them, the overall value of exploring a projection (whether 2D or 3D) is highly limited. As a second finding, we have evidence that 3D projections do indeed bring some added value *vs* their 2D counterparts. More interestingly, we found that such evidence cannot be traced back to classical projection-quality metrics, but rather resides in aspects (of users evaluating such projections) which cannot be quantified by such metrics. This is, we believe, a very interesting finding which motivates a deeper, and potentially different, way of looking at projections to assess their quality and/or added value, beyond the 'bare' quality metrics used nowadays.

Several directions for future work can be outlined based on our results, as follows.

Concerning the 3D skeleton-based rotation mechanism proposed in Chapter 3: We argue that we just 'scratched the surface' of what is possible in this direction. Numerous extensions can be envisaged, *e.g.*, using more cues from the actual data to infer better 3D rotation axes (such as shading and depth gradients); for point clouds, using volume rendering and kernel density estimation (KDE) methods to locate 'dense' struc-

tures in the data and compute local 3D skeletons from those, *e.g.*, using isosurfacing methods. More widely, there is a whole gamut of possibilities of reducing a rendered image (of a 3D structure) to local axes that can serve for rotation, based on heuristics or domain knowledge, that can be explored.

Concerning our explanation of DR scatterplots (Chapters 5 and 6), numerous extensions are also possible. Adding more explanation types, such as inverse correlation, correlation of more than two dimensions, or the presence of specific *n*D data patterns, is a low hanging fruit. One can compute, in parallel, a wide range of local explanations based on a pattern library, and next show the most salient ones in the final view, thereby enriching the current contribution, variance, correlation, and dimensionality views. This would perform a scagnostics-like (Wilkinson et al., 2005) local analysis of the projection, but using patterns described by the high-dimensional data rather than by the scatterplot. Computing a hierarchical explanation, where projection regions are recursively split by additional explanations, is another direction we aim to pursue.

Finally, we argue, at this point we believe convincingly, that the exploration of low-dimensional and high-dimensional datasets visualized by means of 3D metaphors are not fundamentally different techniques. Rather, there is a continuum between the two. Our 3D viewpoint manipulation techniques (proposed for spatial low-dimensional data) could be, indeed, adapted and used to explore 3D projections of high-dimensional data. Conversely, our techniques to explain why points in a 3D DR scatterplot of high-dimensional data are related could be also used to explain why points in a spatial, low-dimensional, 3D scatterplot are related. At the highest level, and as the final conclusion of this thesis, we believe that methods that *unify* the exploration and explanation of data (of whichever dimensionality or spatiality) are the way to go forward.

A. Telea. Real-time 2D skeletonization using CUDA, 2019. http://www.staff.science.uu.nl/$\sim$telea001/Shapes/CUDASkel.

D. K. Agrafiotis. Stochastic proximity embedding. *Journal of Computational Chemistry*, 24(10):1215–1221, 2003.

G. Albuquerque, M. Eisemann, and M. Magnor. Perception-based visual quality measures. In *Proc. IEEE VAST*, pages 11–18, 2011.

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

R. Bade, F. Ritter, and B. Preim. Usability comparison of mouse-based interaction techniques for predictable 3D rotation. In *Proc. Smart Graphics (SG)*, pages 138–150, 2005.

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.

E. J. Beh and C. I. Holdsworth. A visual evaluation of a classification method for investigating the psysicochemical properties of Portugese wine. *Current Anal Chem*, 8(2):205–217, 2012.

M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proc. NIPS*, pages 585–591, 2002.

R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

B. Belousov. Difference between two rotation matrices, 2016. http://www.boris-belousov.net/2016/12/01/quat-dist.

K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *Proc. Intl. Conf. on Database Theory*, pages 217–235. Springer, 1999.

S. Bian, A. Zheng, E. Chaudhry, L. You, and J. J. Zhang. Automatic generation of dynamic skin deformation for animated characters. *Symmetry*, 10(4):89, 2018.

H. Blum. A transformation for extracting new descriptors of shape. In *Models for the Perception of Speech and Visual Form*, pages 362–381. MIT Press, 1967.

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

A. Buja, D. Cook, and D. F. Swayne. Interactive high-dimensional data visualization. *Journal of Computational and Graphical Statistics*, 5(1): 78–99, 1996.

K. Bunte, M. Biehl, and B. Hammer. A general framework for dimensionality reducing data visualization mapping. *Neural Computation*, 24(3):771–804, 2012.

T. T. Cao, K. Tang, A. Mohamed, and T. S. Tan. Parallel banding algorithm to compute exact distance transform with the GPU. In *Proc. ACM SIGGRAPH Symp. on Interactive 3D Graphics and Games*, pages 83–90, 2010.

Y. Chan, C. Correa, and K. L. Ma. Regression cube: a technique for multidimensional visual exploration and interactive pattern finding. *ACM Trans Interact Intell Syst*, 4(1), 2014.

M. Chaouch and A. Verroust-Blondet. Alignment of 3D models. *Graphical Models*, 71(2):63–76, 2009.

M. Chen, S. Mountford, and A. Sellen. A study in interactive 3D rotation using 2D control devices. *Comput Graph Forum*, 22(4):121–129, 1998.

N. Cliff. The eigenvalues-greater-than-one rule and the reliability of components. *Psychological Bulletin*, 103(2):276–279, 1988.

R. R. Coifman and S. Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

D. Coimbra, R. Martins, T. Neves, A. Telea, and F. Paulovich. Explaining three-dimensional dimensionality reduction plots. *Information Visualization*, 15(2):154–172, 2016.

N. Cornea, D. Silver, and P. Min. Curve-skeleton properties, applications, and algorithms. *IEEE TVCG*, 13(3):597–615, 2007.

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, 2009a.

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, 2009b. https://archive.ics.uci.edu/ml/datasets/wine+quality.

L. Costa and R. Cesar. *Shape analysis and classification*. CRC Press, 2000.

J. Cunningham and Z. Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *JMLR*, 16:2859–2900, 2015.

S. Dasgupta. Experiments with random projection. In *Proc. UAI*, pages 143–151. Morgan Kaufmann, 2000.

V. De Silva and J. B. Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Stanford University, 2004.

D. L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc. of the National Academy of Sciences*, 100(10):5591–5596, 2003.

D. van Driel, X. Zhai, Z. Tian, and A. Telea. Enhanced attribute-based explanations of multidimensional projections. In *Proc. EuroVA*, pages 37–41. Eurographics, 2020.

J. Dubinski. When galaxies collide. *Astronomy Now*, 15(8):56–58, 2001.

K. L. Duffin and W. A. Barrett. Spiders: A new user interface for rotation and visualization of N-dimensional point sets. In *Proc. IEEE Visualization*, pages 205–211, 1994.

D. Eler, M. Nakazaki, F. Paulovich, D. Santos, G. Andery, M. Oliveira, J. Neto, and R. Minghim. Visual analysis of image collections. *Visual Computer*, 25(10):677–792, 2009.

N. Elmqvist, P. Dragicevic, and J. D. Fekete. Rolling the dice: multidimensional visual exploration using scatterplot matrix navigation. *IEEE TVCG*, 14(8):1141–1148, 2008.

M. Emory and G. Iaccarino. Visualizing turbulence anisotropy in the spatial domain with componentality contours. *Center for Turbulence Research Annual Research Briefs*, pages 123–138, 2014. URL https://web.stanford.edu/group/ctr/ResBriefs/2014/14_emory.pdf.

A. Endert, P. Flaux, and C. North. Semantic interaction for visual text analytics. In *Proc. ACM CHI*, pages 324–333, 2012.

D. Engel, L. Hüttenberger, and B. Hamann. A survey of dimension reduction methods for high-dimensional data analysis and visualization. In *Proc. IRTG Workshop*, volume 27, pages 135–149. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012.

O. Ersoy, C. Hurter, F. Paulovich, G. Cantareiro, and A. Telea. Skeleton-based edge bundling for graph visualization. *IEEE TVCG*, 17(2):2364 – 2373, 2011.

M. Espadoto, R. Martins, A. Kerren, N. Hirata, and A. Telea. Toward a quantitative survey of dimension reduction techniques. *IEEE TVCG*, 27(3):2153–2173, 2019.

R. Etemadpour, R. Motta, J. de Souza Paiva, R. Minghim, M. D. Oliveira, and L. Linsen. Perception-based evaluation of projection methods for multidimensional data visualization. *IEEE TVCG*, 21(1):81–94, 2014.

S. I. Fabrikant. Spatial metaphors for browsing large data archives, 2000. PhD thesis, University of Colorado Boulder.

A. Falcão, J. Stolfi, and R. Lotufo. The image foresting transform: Theory, algorithms, and applications. *IEEE TPAMI*, 26(1):19–29, 2004.

A. Falcao, C. Feng, J. Kustra, and A. Telea. Multiscale 2D medial axes and 3D surface skeletons by the image foresting transform. In P. Saha, G. Borgefors, and G. S. di Baja, editors, *Skeletonization – Theory, Methods, and Applications*. Elsevier, 2017. Ch. 2.

I. K. Fodor. A survey of dimension reduction techniques. Technical report, US Dept. of Energy, Lawrence Livermore National Labs, 2002. Tech. report UCRL-ID-148494.

E. Frokjaer, M. Hertzum, and K. Hornbaek. Measuring usability: Are effectiveness, efficiency, and satisfaction really correlated? In *Proc. CHI*, pages 345–352, 2000.

X. Geng, D. Zhan, and Z. Zhou. Supervised nonlinear dimensionality reduction for visualization and classification. *IEEE Trans Syst Man Cybern*, 35(6):1098–1107, 2005.

A. Gisbrecht and B. Hammer. Data visualization by nonlinear dimensionality reduction. *WIREs Data Mining Knowledge Discovery*, 5:51–73, 2015.

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

J. C. Gower and D. J. Hand. *Biplots*, volume 54. CRC Press, 1995.

M. Greenacre. *Biplots in practice*. Fundacion BBVA, Bilbao, 2010.

J. Guo, Y. Wang, P. Du, and L. Yu. A novel multi-touch approach for 3D object free manipulation. In *Proc. AniNex*, pages 159–172. Springer, 2017.

N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions, 2009. arXiv:0909.4061 [math.NA].

M. Hancock, S. Carpendale, and A. Cockburn. Shallow-depth 3D interaction: design and evaluation of one-, two- and three-touch techniques. In *Proc. ACM CHI*, pages 1147–1156, 2007.

M. Hancock, T. ten Cate, S. Carpendale, and T. Isenberg. Supporting sandtray therapy on an interactive tabletop. In *Proc. ACM CHI*, pages 2133–2142, 2010.

X. He and P. Niyogi. Locality preserving projections. In *Proc. NIPS*, pages 153–160, 2004.

X. He, D. Cai, S. Yan, and H. Zhang. Neighborhood preserving embedding. In *Proc. IEEE ICCV*, pages 1208–1213, 2005.

K. Henriksen, J. Sporring, and K. Hornbaek. Virtual trackballs revisited. *IEEE TVCG*, 10(2):206–216, 2004a.

K. Henriksen, J. Sporring, and K. Hornbaek. Virtual trackballs revisited – source code, 2004b. http://image.diku.dk/research/trackballs/index.html.

W. H. Hesselink and J. B. T. M. Roerdink. Euclidean skeletons of digital image and volume data in linear time by the integer medial axis transform. *IEEE TPAMI*, 30(12):2204–2217, 2008.

N. Heulot, J. D. Fekete, and M. Aupetit. Visualizing dimensionality reduction artifacts: An evaluation, 2017. arXiv:1705.05283v1 [cs.HC].

K. Hinckley, J. Tullio, R. Pausch, D. Proffitt, and N. Kassell. Usability analysis of 3D rotation techniques. In *Proc. UIST*, pages 1–10, 1997.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

P. Hoffman and G. Grinstein. A survey of visualizations for high-dimensional data mining. *Information Visualization in Data Mining and Knowledge Discovery*, 104:47–82, 2002.

H. Hotelling. Analysis of a complex of statistical variables into principal components. *J Edu Pysiol*, 24:417–441; 498–520, 1933.

J. Hultquist. A virtual trackball. In *Graphics Gems*, volume 1, pages 462–463, 1990.

A. Hyvarinen. Fast ICA for noisy data using Gaussian moments. In *Proc. IEEE ISCAS*, volume 5, pages 57–61, 1999.

A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multi-dimensional geometry. In *Proc. IEEE VIS*, pages 361–378, 1990.

J. Dubinski *et al.* GRAVITAS: Portraits of a universe in motion, 2006. https://www.cita.utoronto.ca/~dubinski/galaxydynamics/gravitas.html.

B. Jackson, T. Y. Lau, D. Schroeder, K. C. Toussaint, and D. F. Keefe. A lightweight tangible 3D interface for interactive visualization of thin fiber structures. *IEEE TVCG*, 19(12):2802–2809, 2013.

B. Jacob and G. Guennebaud. Eigen numerical library, 2020. http://eigen.tuxfamily.org.

I. Jacob and J. Oliver. Evaluation of techniques for specifying 3D rotations with a 2D input device. In *Proc. HCI*, pages 63–76, 1995.

P. Joia, D. Coimbra, J. A. Cuminato, F. V. Paulovich, and L. G. Nonato. Local affine multidimensional projection. *IEEE TVCG*, 17(12):2563–2571, 2011.

I. T. Jolliffe. Principal component analysis and factor analysis. In *Principal Component Analysis*, pages 115–128. Springer, 1986.

I. T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

D. Kaye and I. Ivrissimtzis. Mesh alignment using grid based PCA. In *Proc. CGTA)*, pages 174–181, 2015.

J. Kehrer and H. Hauser. Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE TVCG*, 19(3):495–513, 2013.

Keras repository. Reuters dataset, 2021. https://keras.io/api/datasets/reuters.

J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

J. Kustra, A. Jalba, and A. Telea. Probabilistic view-based curve skeleton computation on the GPU. In *Proc. VISAPP*, pages 237–246. SCITEPRESS, 2013.

J. Kustra, A. Jalba, and A. Telea. Robust segmentation of multiple intersecting manifolds from unoriented noisy point clouds. *Comp Graph Forum*, 33(4):73–87, 2014.

D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Proc. NIPS*, pages 556–562, 2001.

J. A. Lee and M. Verleysen. Quality assessment of dimensionality reduction: Rank-based criteria. *Neurocomputing*, 72(7):1431–1443, 2009.

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

M. Lichman. UCI machine learning repository, 2013. http://archive.ics.uci.edu/ml.

S. Liu, D. Maljovec, B. Wang, P. T. Bremer, and V. Pascucci. Visualizing high-dimensional data: Advances in the past decade. *IEEE TVCG*, 23 (3):1249–1268, 2015.

S. Liu, M. Zhang, P. Kadam, and C. C. J. Kuo. *3D Point Cloud Analysis: Traditional, Deep Learning, and Explainable Machine Learning Methods*. Springer, 2021.

M. Livesu, F. Guggeri, and R. Scateni. Reconstructing the curve-skeletons of 3D shapes using the visual hull. *IEEE TVCG*, 18(11):1891–1901, 2012.

W. Lueks, A. Gisbrecht, and B. Hammer. Visualizing the quality of dimensionality reduction. *Neurocomputing*, 112:109–123, 2013.

S. Marschner and P. Shirley. *Fundamentals of Computer Graphics.* CRC Press, 2021.

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

R. Martins, R. Minghim, and A. C. Telea. Explaining neighborhood preservation for multidimensional projections. In *Proc. CGVC*, pages 121–128. Eurographics, 2015.

L. McInnes, J. Healy, and J. Melville. UMAP: Uniform manifold approximation and projection for dimension reduction, 2018. arXiv:1802.03426v2 [stat.ML].

A. Meijster, J. Roerdink, and W. Hesselink. A general algorithm for computing distance transforms in linear time. In *Mathematical Morphology and its Applications to Image and Signal Processing*, pages 331–340. Springer, 2002.

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 *Proc. Brazilian Symposium on Software Engineering (SBES)*, pages 11–20, 2010.

R. Motta, R. Minghim, A. Lopes, and M. Oliveira. Graph-based measures to assist user assessment of multidimensional projections. *Neurocomputing*, 150:583–598, 2015.

T. Munzner. *Visualization Analysis and Design: Principles, Techniques, and Practice.* CRC Press, 2014.

NASA. Example of specifying 3d rotations around coordinate axes for a rocket system, 2022. https://www.grc.nasa.gov/www/k-12/rocket/rotations.html.

G. Newby. Empirical study of a 3D visualization for information retrieval tasks. *J Intell Inf Syst*, 18(1):31–53, 2002.

L. G. Nonato and M. Aupetit. Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. *IEEE TVCG*, 25(8):2650–2673, 2018.

G. R. North, T. L. Bell, R. F. Cahalan, and F. J. Moeng. Sampling errors in the estimation of empirical orthogonal functions. *Mon Weather Rev*, 110:699–706, 1982.

L. J. O'Donnell and C. F. Westin. An introduction to diffusion tensor image analysis. *Neurosurg Clin N Am*, 22(2):185–196, 2011.

S. Oeltze, H. Doleisch, and H. Hauser. Interactive visual analysis of perfusion data. *IEEE TVCG*, 13(6):1392–1399, 2007.

R. L. Ogniewicz and O. Kubler. Hierarchic Voronoi skeletons. *Patt Recog*, (28):343–359, 1995.

K. Olsen, R. Korfhage, and K. Sochats. Visualization of a document collection: the VIBE system. *Inform Process Manag*, 29(1):69–81, 1993.

P. B. P and A. Falguerolles. Application of resampling methods to the choice of dimension in principal component analysis. In *Computer Intensive Methods in Statistics*, pages 167–176. Springer, 1993.

P. Pagliosa, F. Paulovich, R. Minghim, H. Levkowitz, and L. Nonato. Projection inspector: Assessment and synthesis of multidimensional projections. *Neurocomputing*, 150:599–610, 2015.

T. Partala. Controlling a single 3D object: Viewpoint metaphors, speed, and subjective satisfaction. In *Proc. INTERACT*, pages 536–543, 1999.

F. V. Paulovich, L. G. Nonato, R. Minghim, and H. Levkowitz. Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. *IEEE TVCG*, 14 (3):564–575, 2008.

F. V. Paulovich, F. Toledo, G. Telles, R. Minghim, and L. G. Nonato. Semantic wordification of document collections. *Computer Fraphics Forum*, 31(3):1145–1153, 2012.

K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572, 1901.

H. Piringer, R. Kosara, and H. Hauser. Interactive F + C visualization with linked 2D/3D scatterplots. In *Proc. IEEE CMV*, pages 49–60, 2004.

J. Poco, R. Etemadpour, F. V. Paulovich, T. Long, P. Rosenthal, M. C. F. Oliveira, L. Linsen, and R. Minghim. A framework for exploring multidimensional data with 3D projections. *Comput Graph Forum*, 30(3): 1111–1120, 2011.

R. Rao and S. K. Card. The table lens: Merging graphical and symbolic representations in an interactive focus+context visualization for tabular information. In *Proc. ACM SIGCHI*, pages 318–322, 1994.

P. Rauber, R. da Silva, S. Feringa, M. Celebi, A. Falcao, and A. Telea. Interactive image feature selection aided by dimensionality reduction. In *Proc. EuroVA*, pages 97–101, 2015.

D. Reniers, J. J. van Wijk, and A. Telea. Computing multiscale skeletons of genus 0 objects using a global importance measure. *IEEE TVCG*, 14 (2):355–368, 2008.

C. Richter. *Designing Flexible Object-Oriented Systems with UML*. New Riders Publishing, 1999.

F. C. M. Rodrigues, M. Espadoto, R. Hirata, and A. Telea. Constructing and visualizing high-quality classifier decision boundary maps. *Information*, 10(9):280–297, 2019.

R. S. V. Rodrigues, J. F. M. Morgado, and A. J. P. Gomes. Part-based mesh segmentation: A survey. *Comp Graph Forum*, 37(6):235–274, 2018.

A. Rosenfeld and J. Pfaltz. Distance functions in digital pictures. *Pattern Recognition*, 1:33–61, 1968.

D. A. Ross, J. Lim, R. S. Lin, and M. H. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77 (1-3):125–141, 2008.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

H. Sanftmann and D. Weiskopf. Illuminated 3D scatterplots. *Comput Graph Forum*, 28(3):642–651, 2009.

H. Sanftmann and D. Weiskopf. 3D scatterplot navigation. *IEEE TVCG*, 18(11):1969–1978, 2012.

B. Schölkopf, A. Smola, and K. R. Müller. Kernel principal component analysis. In *Proc. ICANN*, pages 583–588. Springer, 1997.

T. Schreck, T. von Landesberger, and S. Bremm. Techniques for precision-based visual analysis of projected data. *Information Visualizationhes*, 9(3):181–193, 2010.

M. Sedlmair and M. Aupetit. Data-driven evaluation of visual quality measures. *Comp Graph Forum*, 34(3):545–559, 2015.

M. Sedlmair, T. Munzner, and M. Tory. Empirical guidance on scatterplot and dimension reduction technique choices. *IEEE TVCG*, pages 2634–2643, 2013.

K. Shoemake. Arcball: A user interface for specifying three-dimensional orientation using a mouse. In *Proc. Graphics Interface*, 1992.

K. Siddiqi and S. Pizer. *Medial Representations: Mathematics, Algorithms and Applications*. Springer, 2008.

R. da Silva. *Visualizing Multidimensional Data Similarities – Improvements and Applications*. PhD thesis, University of Groningen, Netherlands, 2016.

R. da Silva, P. Rauber, R. Martins, R. Minghim, and A. Telea. Attribute-based visual explanation of multidimensional projections. In *Proc. EuroVA*, pages 97–101, 2015.

M. Sips, B. Neubert, J. Lewis, and P. Hanrahan. Selecting good views of high-dimensional data using class consistency. *Comp Graph Forum*, 28(3):831–838, 2009.

A. Sobiecki, H. Yasan, A. Jalba, and A. Telea. Qualitative comparison of contraction-based curve skeletonization methods. In *Proc. ISMM*. Springer, 2013.

C. Sorzano, J. Vargas, and A. Pascual-Montano. A survey of dimensionality reduction techniques, 2014. arXiv:1403.2877 [stat.ML].

A. Tagliasacchi, T. Delame, M. Spagnuolo, N. Amenta, and A. Telea. 3D skeletons: A state-of-the-art report. *Comp Graph Forum*, 35(2):573–597, 2016.

J. Tang, J. Liu, M. Zhang, and Q. Mei. Visualizing large-scale and high-dimensional data. In *Proc. WWW*, pages 287–297, 2016.

J. W. H. Tangelder and R. C. Veltkamp. A survey of content based 3D shape retrieval methods. *Multimedia Tools and Applications*, 39(441): 441–471, 2008.

A. Tatu, P. Bak, E. Bertini, D. Keim, and J. Schneidewind. Visual quality metrics and human perception: An initial study on 2D projections of large multidimensional data. In *Proc. AVI*, pages 49–56. ACM, 2010.

M. Tavanti and M. Lind. 2D *vs* 3D, implications on spatial memory. In *Proc. IEEE InfoVis*, pages 139–145, 2001.

A. Telea. Feature preserving smoothing of shapes using saliency skeletons. In *Proc. VMLS*, pages 136–148. Springer, 2011.

A. Telea. Source code for salience skeleton computation, 2014a. https://webspace.science.uu.nl/$\sim$telea001/Shapes/Salience.

A. Telea and A. Jalba. Voxel-based assessment of printability of 3D shapes. In *Proc. ISMM*. Springer, 2011.

A. C. Telea. Combining extended table lens and treemap techniques for visualizing tabular data. In *Proc. EuroVis*, pages 120–127, 2006.

A. C. Telea. *Data Visualization – Principles and Practice*. CRC Press / Taylor and Francis, 2014b. 2^{nd} edition.

A. Telea and J. J. van Wijk. An augmented fast marching method for computing skeletons and centerlines. In *Proc. VisSym*, pages 251–259. Springer, 2002.

J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290 (5500):2319–2323, 2000.

TensorFlow. Embedding projector tool, 2021. https://projector.tensorflow.org.

Z. Tian, X. Zhai, G. van Steenpaal, L. Yu, E. Dimara, M. Espadoto, and A. Telea. Comparing 2D and 3D explained projections – evaluation datasets, metrics, and results, 2021a. https://tianzonglin.github.io/project-compare.

Z. Tian, X. Zhai, G. van Steenpaal, L. Yu, E. Dimara, M. Espadoto, and A. Telea. Explanatory visualization tool for 2D and 3D projections, 2021b. Source code https://git.science.uu.nl/vig/mscprojects/Pointctl.

Z. Tian, X. Zhai, D. van Driel, G. van Steenpaal, M. Espadoto, and A. Telea. Using multiple attribute-based explanations of multidimensional projections to explore high-dimensional data. *Computers & Graphics*, 2021c. https://doi.org/10.1016/j.cag.2021.04.034.

Z. Tian, X. Zhai, G. van Steenpaal, L. Yu, E. Dimara, M. Espadoto, and A. Telea. Quantitative and qualitative comparison of 2d and 3d projection techniques for high-dimensional data. *Information*, 12(6), 2021d. https://doi.org/10.3390/info12060239.

W. S. Torgerson. *Theory and Methods of Scaling*. Wiley, 1958.

L. van der Maaten and G. E. Hinton. Visualizing data using t-sne. *JMLR*, 9:2579–2605, 2008.

L. van der Maaten and E. Postma. Dimensionality reduction: A comparative review. Technical report, Tilburg University, Netherlands, 2009. Tech. report TiCC TR 2009-005.

J. Venna and S. Kaski. Visualizing gene interaction graphs with local multidimensional scaling. In *Proc. ESANN*, pages 557–562, 2006a.

J. Venna and S. Kaski. Visualizing gene interaction graphs with local multidimensional scaling. In *Proc. ESANN*, pages 557–562, 2006b.

S. D. Vito, E. Massera, M. Piga, L. Martinotto, and G. D. Francia. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B: Chemical*, 129 (2):750–757, 2008. https://archive.ics.uci.edu/ml/datasets/Air+Quality.

S. Westerman and T. Cribbin. Mapping semantic information in virtual space: dimensions, variance and individual differences. *Int J Hum Comput St*, 53(5):765–787, 2000.

S. Westerman, J. Collins, and T. Cribbin. Browsing a document collection represented in two- and three- dimensional virtual information space. *Int J Hum Comput St*, 62(6):713–736, 2005.

J. J. van Wijk and A. Telea. Enridged contour maps. In *Proc. IEEE Visualization*, pages 69–74, 2001.

L. Wilkinson, A. Arland, and R. Grossman. Graph-theoretic scagnostics. In *Proc. InfoVis*, pages 157–164, 2005.

W. H. Wolberg, W. N. Street, and O. L. Mangasarian. Wisconsin breast cancer dataset, 2021. https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic).

S. Wu, B. Li, J. Yang, and S. Shukla. Predictive modeling of high-performance concrete with regression analysis. In *Proc. IEEE Intl. Conf. on Industrial Engineering and Engineering Management*, 2010.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, 2017. arXiv 1708.07747 [cs.LG].

H. Xie, J. Li, and H. Xue. A survey of dimensionality reduction techniques based on random projection, 2017. arXiv:1706.04371 [cs.LG].

A. Yates, A. Webb, M. Sharpnack, H. Chamberlin, K. Huang, and R. Machiraju. Visualizing multidimensional data with glyph SPLOMs. *Computer Graphics Forum*, 33(3):301–310, 2014.

I. C. Yeh and C. H. Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2):2473–2480, 2009. https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients.

I. C. Yeh. Modeling of strength of high performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12):1797–1808, 1998a.

I. C. Yeh. Modeling of strength of high performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(2):1797–1808, 1998b. https://archive.ics.uci.edu/ml/datasets/concrete+compressive+strength.

J. Yi, R. Melton, and J. Stasko. Dust & magnet: multivariate information visualization using a magnet metaphor. *Inform Visual*, 4(4):239–256, 2005.

H. Yin. Nonlinear dimensionality reduction and data visualization: A review. *Intl. Journal of Automation and Computing*, 4(3):294–303, 2007.

L. Yu and T. Isenberg. Exploring one- and two-touch interaction for 3D scientific visualization spaces. In *Posters of Interactive Tabletops and Surfaces (Proc. ITS)*, 2009.

L. Yu, P. Svetachov, P. Isenberg, M. H. Everts, and T. Isenberg. FI3D: Direct-touch interaction for the exploration of 3D scientific visualization spaces. *IEEE TVCG*, 16(6):1613–1622, 2010.

L. Yu, K. Efstathiou, P. Isenberg, and T. Isenberg. Efficient structure-aware selection techniques for 3D point cloud visualizations with 2DOF input. *IEEE TVCG*, 18(12):2245–2254, 2012.

J. Yuan, S. Xiang, J. Xia, L. Yu, and S. Liu. Evaluation of sampling methods for scatterplots. *IEEE TVCG*, 27(2):1720–1730, 2021.

L. Zeng. The wine dataset analysis, 2021. https://rpubs.com/Li2019/Wine.

X. Zhai, L. Yu, X. Chen, and A. Telea. Source code and videos of interactive skeleton-based axis rotation, 2019. http://www.staff.science.uu.nl/~telea001/Shapes/CUDASkelInteract.

X. Zhai, X. Chen, L. Yu, and A. Telea. Interactive axis-based 3D rotation specification using image skeletons. In *Proc. 15^{th} International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications – Volume 1: GRAPP*, pages 169–178. SciTePress, 2020.

X. Zhai, L. Yu, X. Chen, and A. Telea. Skeleton-and-trackball interactive rotation specification for 3D scenes. In *Communication in Computer and Information Science*, pages 26–52. Springer, 2022.

S. Zhang, B. Guo, A. Dong, J. He, Z. Xu, and S. Chen. Cautionary tales on air-quality improvement in Beijing. *Proc Royal Society A*, 473(2205):170–187, 2017. https://archive.ics.uci.edu/ml/datasets/Beijing+Multi-Site+Air-Quality+Data.

T. Zhang, J. Yang, D. Zhao, and X. Ge. Linear local tangent space alignment and application to face recognition. *Neurocomputing*, 70(7-9): 1547–1553, 2007.

Z. Zhang and J. Wang. MLLE: Modified locally linear embedding using multiple weights. In *Proc. NIPS*, pages 1593–1600, 2007.

Z. Zhang and H. Zha. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM journal on scientific computing*, 26(1):313–338, 2004.

Y. J. Zhao, D. Shuralyov, and W. Stuerzlinger. Comparison of multiple 3D rotation methods. In *Proc. IEEE VECIMS*, pages 19–23, 2011.

H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006.

BIOGRAPHY

Xiaorui (Rui) Zhai was born on March 10, 1982 in Henan, China. She followed the courses of the Hangzhou Dianzi University to obtain first her Bachelor degree in July 2004, followed by her Master degree at the same university in March 2007, which she ended with a thesis on motion segmentation and shadow detection. Following graduation, she obtained in July 2007 a lecturer position at the Jiyang College of the Zhejiang Agriculture and Forestry University (ZAFU) in China, where she started teaching various courses in computer science and data science including computer graphics and digital signal processing. In 2018, she was the only student to obtain a scholarship from ZAFU to pursue a PhD study abroad. Following this, she chose to pursue her PhD research at the Scientific Visualization and Computer Graphics (SVCG) group at the Bernoulli Institute, University of Groningen, which led to this PhD thesis. Her general research interests span the intersection of 2D and 3D data visualization, interaction techniques, computer graphics, and image processing.

ACKNOWLEDGMENTS

I am, above all, sincerely thankful to the support provided by the Jiyang College of the Zhejiang Agriculture and Forestry University (ZAFU) for pursuing a PhD study abroad. This support not only covered the financial expenses related to my stay in the Netherlands, travel to and from the Netherlands to China, but also kept the opportunity for me to return to the same teaching position I had at the university after completing my PhD study, therefore offering a clear career path to follow after my PhD. Without this comprehensive support, driven by my university's vision to further develop its researchers by obtaining PhD degrees abroad, I would have likely not been able to conduct research leading to a PhD degree.

Most importantly next, I want to express my sincerest thanks to my mentor prof. dr. Alex Telea who gave the most help to me in all fields. His smile, encouraging words, professional knowledge, vision and sincerity, have deeply helped me and gave me courage. He is my hero. I also want to express my sincerest thanks to my mentor dr. Lingyun Yu who was also my classmate and roommate during my undergraduate and postgraduate studies. She is my best friend and my sweet mentor who helped me a lot in my research.

Thanks also go to the members of the SVCG research group: Prof. dr. J.B.T.M. Roerdink, Prof. dr. J. Kosinka, Xingyu Chen, Jieying Wang, and Gerben Hettinga. And thanks to the members of the VIG research group at Utrecht University: Michael Behrisch, Tamara Mtsentlintze, and Zonglin Tian. All of them helped me a lot during my PhD period.

Last but not least, I must show my gratitude to my husband Xiaojun Tong and my son Haoxiang Tong. Both of you gave me the courage and support to study abroad. Love you so much and forever. I dedicate this manuscript to both of you.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Final Version as of September 28, 2022 (`classicthesis`).