Innovations in High Dimensional Data Exploration, Representation and Generation

A Dissertation Presented

by

Bing Wang

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

August 2016

Stony Brook University

The Graduate School

Bing Wang

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation.

Klaus Mueller – Dissertation Advisor
Professor, Stony Brook University

Arie Kaufman - Chairperson of Defense
Distinguished Professor and Chairman, Stony Brook University

Xianfeng Gu
Associate Professor, Stony Brook University

Michael Grossberg
Assistant Professor, Computer Science, The City College of New York

This dissertation is accepted by the Graduate School

Nancy Goroff

Interim Dean of the Graduate School

Abstract of the Dissertation

Innovations in High Dimensional Data Exploration, Representation and Generation

by

Bing Wang

Doctor of Philosophy

in

Computer Science

Stony Brook University

2016

Data with many attributes have become commonplace in a wide range of domains. In these data, the most interesting relations are often multivariate and are generally confusing to most people. Efforts have been made to design proper tools to recognize those high dimensional relationships reliably but those tools are often far off from making use of the innate 3D scene understanding capabilities of the human visual system.

We present a framework that eases this barrier by design, called the *Subspace Voyager*. It decomposes a high-dimensional data space into a continuum of generalized 3D subspaces. Analysts can then explore these 3D subspaces individually via the familiar trackball interface and use additional facilities to smoothly transition to adjacent subspaces for expanded space comprehension.

On top of the *Subspace Voyager*, we propose a novel 3D shaded shape representation for non-spatial data. This representation visualizes data matrices in the most natural 3D forms that include depth cues, such as occlusion, shading, perspective distortion, shadows, and so on. Our user study suggests that mainstream users prefer shaded displays over scatterplots for visual cluster analysis tasks. And further, our experiments also provide evidence that 3D displays can better communicate spatial relationships, size, and shape of clusters.

When designing those tools, we often had difficulties acquiring proper testing data. We therefore propose an interactive data generation tool – SketchPad$^{N\text{-}D}$. The core concept in our SketchPad$^{N\text{-}D}$ is WYSIWYG (What You See Is What You Get) because it allows users to generate dataset in the same interface they use to visualize it such that they do not need to switch back and forth between data manipulation and visualization tools.

To my parents Jinfeng Wang and Zhixia Xia

Table of Contents

List of Figures

Acknowledgments

I wish to thank everyone who has helped me in accomplishing this work, especially my advisor Professor Mueller and all my lab mates.

Publications

•**B. Wang**, K. Mueller, “Shape Up: Are Shaded Surface Displays More Appealing and Informative than Scatterplots?” in submission

•**B. Wang**, K. Mueller, “The Subspace Voyager: Exploring High-Dimensional Data along a continuum of Salient 3D Subspaces,” *IEEE Trans. on Visualization and Computer Graphics,* under revision

•E. Papenhausen, **B. Wang**, M. H. Langston, B. Meister, M. Baskaran, T. Henretty, T. Izubuchi, A. Johnson, C. Jung, M. Lin, K. Mueller and R. Lethin, “Polyhedral User Mapping and Assistant Visualizer Tool for the R-Stream Auto-Parallelizing Compiler*,” the 3rd IEEE Working Conference on Software Visualization*

•**B. Wang**, K. Mueller, “Does 3D Really Make Sense for Visual Cluster Analysis? Yes!” *Intern'l Workshop on 3DVis: Does 3D Really Make Sense for Data Visualization? (with VIS 2014).* Paris, France, November, 2014

•S. Cheng, **B. Wang**, Z. Zhang, K. Mueller, “Balanced layouts using the composite data-variable matrix,” *IEEE Conference on Visual Analytics Science and Technology (VAST),* 235-236, 2014

•**B. Wang**, P. Ruchikachorn, K. Mueller, “SketchPad$^{N\text{-}D}$: WYDIWYG Sculpting and Editing in High-Dimensional Space,” *IEEE Trans. on Visualization and Computer Graphics,* Vol. 19, No. 12, December 2013

•E. Papenhausen, **B. Wang**, S. Ha, A. Zelenyuk, D. Imre, K. Mueller, “GPU-Accelerated Incremental Correlation Clustering of Large Data with Visual Feedback*,” The First IEEE Workshop on Big Data Visualization,* Santa Clara, CA, October, 2013

•Z. Zhang, **B. Wang**, F. Ahmed, IV Ramakrishnan, A. Viccellio, R. Zhao, K. Mueller*, “*The Five W’s for Information Visualization with Application to Healthcare Informatics”, *IEEE Transactions on Visualization and Computer Graphics,* 19 (11): 1895-1910, 2013

•P. Ruchikachorn, **B. Wang**, K. Mueller, "SketchPad N-D: An Interface for High-Dimensional Dataset Generation and Editing," *IEEE Visualization, poster session*, Seattle, WA, October 2012

CHAPTER 1

INTRODUCTION

Data with many attributes have become commonplace in a wide range of domains, such as science, business, medicine, etc. In these data, the most interesting relations are often multivariate, and gaining proper tools to recognize these relationships reliably is still an active area of research. While automated analysis can be useful in finding some of the high-dimensional patterns, adding the human into the loop can break ties and can also help discern patterns in confounding and noisy data settings that can benefit from the intricate reasoning faculties of human domain experts. However, we are still far off from having effective visual tools for high-D data analytics that make the best use of the innate capabilities of the human visual system and at the same time also observe its limitations.

The human retina contains about 7M cones and 100M rods. This is roughly equivalent to a 2.5k×2.5k pixel color camera and a 10k×10k pixel grey level camera. Looking at today's technology, the resolution of consumer-level cameras matches or will soon match the resolution of the human eye. But can cameras recognize 3D objects? They surely cannot – even when we assemble two of them into a stereoscopic pair akin to the configuration of the human visual system. This is because the power of the human visual system is not the density of its retinal neurons – it is the massive processing that follows the image acquisition. Or better, the ability to rapidly index and interpolate a vast database of pre-stored 3D models of the world that we have acquired from early childhood on. It was Hermann von Helmholtz [34], who in the 19th century performed the first modern study of visual perception. When von Helmholtz examined the human eye he concluded that they could not aid humans in the perception of 3D shapes directly. And indeed, as was already suspected by Helmholtz, it was later scientifically shown that human perception of the 3D physical world we live in is learned during infancy. During this time an unconscious inferential chain is established which is used to transform the input coming from the eye's optical system into the perception of 3D shape and relations. These neural circuits can not only make inferences about 3D shapes and topologies, but also resolve complex patterns and textures.

This system served us very well for 3D visualization tasks, such as the search for tumors in medical volume renderings or the quest for vortices in a fluid flow volume. But it is overloaded when it is asked to look for features in a high-dimensional (N-D) dataset. The human visual system has just not built the circuitry required to recognize high-dimensional objects. One approach to aid humans in the task of comprehending and navigating high-dimensional space is to serialize it into three-dimensional subspaces. However, the number of such subspaces suffers from combinatorial explosion and so computational guidance is needed to navigate the maze. So are we out of luck to ever see “N-D”? Indeed we may be so with conventional means, but we can take advantage of our complex neural circuitry and try to build visual aids that build on models and concepts we have learnt to use in the past and so feel natural and familiar when applying them to N-D visualization tasks. Stereo vision and motion parallax (objects closer to the viewer moves faster than the object further away) are two great examples. Especially the latter is an interesting concept since we can easily facilitate it on the computer, via interaction, without the need for special glasses.

In this dissertation, we describe a web based framework and interface that is inspired by the above mentioned ideas, called the *Subspace Voyager* [94]. It serializes the exploration of high-D space into a continuous travel along a string of generalized 3D subspaces. This serialization allows us to abolish the complex interactions and representations that are often typical to high-D space exploration tools and replace them with paradigms familiar to most people, such as trackballs, maps, and word clouds. Our interface uses these to help users explore the generalized 3D subspaces, navigate the continuum of 3D subspaces, and assess the relevance of individual attributes for a given subspace, respectively. Our Subspace voyager can also automatically optimizes views in a close range according to user specific criteria, and this provides the users with the views that are truly interesting to them.

The simplicity gained through the 3D subspace decomposition comes at a price – the extent of the transformations defined on such a restricted subspace is limited and may not reach far enough to generate a projection in which a pattern of current interest is well expressed. To enable a reach beyond these limits we have augmented the trackball with extra capabilities that allow users to “chase” the discovered patterns by moving to adjacent 3D subspaces via simple mouse interactions. In this way patterns can be observed that are truly multivariate and not restricted to a single 3D subspace.

We strive to make the interactions our Subspace Voyager offers as familiar and intuitive as possible. For example, we let the users to use simple mouse movement to rotate the trackball, let them to add explored views in a familiar map setting and let them drag and drop previous views back to the main exploration area for further inspection. What’s more, in order to offer the users the ability to present their findings to other people in a continuous fashion, we include a ‘presentation mode’ in our framework that displays the key findings by animating the transient path.

The Subspace Voyager allows us to explore the high-dimensional data and understand high-dimensional relationships by creating motion parallax via interaction. Motion parallax offers an illusion of a third dimension which is used to tell close and distant objects apart. This dimension is what is referred to as ‘depth cue’. Since we already down project the data to a 3D space, can we make use of some other forms of depth cues such that 3D cognition alone, instead of a series of interactions, to analyze the data? The problem with high-dimensional data is that they lack this kind of depth cues which allow us to appreciate shapes in 3D volume renderings.

We tackle this problem by using the Marching Cubes algorithm [60] to extract the 3D shapes of the data density fields and render these shapes using advanced computer graphics techniques, instead of operating on the data points directly. We propose a novice non-spatial data representation – the *3D Shaded Shape Representation* [93]. We also use other depth and shape cues, such as shading, shadows, depth of field, transparency, and the like, and control them via interaction.

Information visualization deals mostly with non-spatial data which can be multivariate. So one might say that adding a third dimension for the visualization of these data only offers insignificant gains when the number of variables is on the order of 10s, 100s, or even 1,000s. While this is a valid argument, it is not the reason why we advocate for 3D graphics. Rather, we advocate for it since, as the wide popularity of 3D movies and 3D shaded bar and pie charts readily show, 3D graphics is more appealing and potentially more engaging to general users than plain 2D graphics. And furthermore, due to the innate 3D reasoning capabilities of the human visual system, 3D renderings can also potentially facilitate better data understanding. We successfully tested both of these hypotheses for the research through former user study.

One problem we realized when testing the Subspace Voyager was that it is sometimes hard to acquire real world data that fully meets the evaluation goals and artificial data specifically designed for certain testing purpose would be preferred. What's more, even real datasets could be found, editing would still be recommended before bringing them into the testing circle. Proficient users could use statistical languages, scripts or software such as R or Matlab to generate data. But this is a tedious procedure and they need to be very careful the entire time. Any small mistake could lead to an entirely wrong dataset and without much visual aid, fixing it could be very difficult. For general users, this task would even feel like mission impossible. There are some tools for users to design data, but to our knowledge there's no such tools for high dimensional data generation.

Hence, we propose a sketching based data generation tool – *SketchPad*$^{N\text{-}D}$ [95]. It's based on the dynamic scatterplot interface [60] and allows users to simply sketch out what they want on a 2D plane. A 2D intermediate dataset is generated based on this sketching. Such a plane can be either axis-aligned or non-axis-aligned. For the former, we randomize the values for the undefined dimensions and combine them with the 2D intermediate data. For the latter, we build a new coordinate system based on the two selected non-axis-aligned axes using Gram Schmidt Orthonormalization process [31], treat it the same as the axis-aligned case, generate the data and then rotate the data back to the original data space. After this data generation step, users can modify the initial data, such as brushing out non needed parts or fixing editing errors, and use either a scatterplot matrix (for axis-aligned case) or a series of user selected arbitrary views (for non-axis-aligned case), to monitor the effect.

The main advantage of this tool is the commonly separated two steps - data generation and data visualization – are tightly linked here. This would give users more control over what is going on and thus more rapid data generation. Users could also use the same interface to edit data – even real world ones.

The remainder of this dissertation is organized as follows. Chapter 2 reports on background information and related work. Chapter 3 presents a study we’ve conducted on the intrinsic dimensionality. Chapter 4 presents a new interactive data analysis tool that allows users to traverse between salient subspaces in the high dimensional space. Chapter 5 describes a novel non-spatial data representation which fully utilizes 3D rendering techniques and feels more natural and appealing to general users. Chapter 6 introduces a sketching based high dimensional data generation tool where users can design the data in the same interface where it is also visualized. Chapter 7 summarizes our works and points to the directions of the future work.

CHAPTER 2

BACKGROUND

Analyzing high dimensional data is an inherently difficult task. We human beings are not trained for reasoning with anything that has more than three dimensions and a lot of well performing techniques in low dimensional space would probably not working in high-dimensional space. The phrase ‘curse of dimensionality’ [11] vividly depicts this phenomenon and it also describes the fact that limitation of the variance of the distribution of pairwise distance versus the minimum pairwise distance goes to zero as the dimensionality grows to infinity [2][25]. Therefore, when dealing with high dimensional data, extra effort should be made.

Visualization has always been a good tool for data analysis, especially high dimensional ones. Visualization has been defined as “the use of computer-supported, interactive visual representations of data to amplify cognition” [13] and “The purpose of visualization is insight, not pictures”. It is more intuitive looking at a picture and deducts some knowledge from it than barely looking at rigid numbers. When visualization is integrated into the whole reasoning chain, this process is call “Visual Analytics”.

Visual Analytics dates back to the 18th century when Dr. John Snow used a map to visualize the outbreak of Cholera [120]. The newer applications of visual analytics are ubiquitous, such as business intelligence [84], health care [73][104], finance [24] and so on. Kang [42] et al evaluated how visual analytics can help investigative analysis. Keim et at. [43][44][46] in their books described in detail what visual analytics is, what methods should be used, what its application areas are and what the challenges are. To summarize, in order to analyze data visually, the user has to have the required domain knowledge, to collect the data, to assimilate the data, to visualize the data and as the last step, to do computational analysis. The three main tasks are clear –visualize the data, analyze the data and obtain the data. For high dimensional data, the challenges are: what visualization paradigm should we use that best represents the underlying features of this data? What techniques we should apply that best facilitate the users when reasoning with the data given its visualization? If we have designed a good visual analytics tool but there is no testing data available, how can we easily generate high dimensional data?

2.1 VISUALIZE HIGH DIMENSIONAL DATA

There are various visualization techniques that have been applied in the high dimensional space. *Parallel coordinates* [38] is a famous one. It represents the data points as polylines crossing a series of parallel axes. This essentially unrolls the high-dimensional space into a serialization of axis pairs and so gives good visual access onto the space. However, a shortcoming of parallel coordinates is that over plotting can become a significant problem once the number of data points grows. And also, the ordering of axes can have a significant influence what patterns can be seen, especially when it comes to correlation [39][102]. *Radviz* [35] is a non-linear multi-dimensional visualization that embeds points inside a circle. Its position is decided using physical method: suppose there is a sprint coming from each dimension holding the point, where it would finally stops. The *Self Organizing M*ap [51] not only reduces the dimensionality of data, but also maps the similar items together. It is actually a neural network method, and its organization is achieved by competing for the representation of the samples. *Star coordinates* [41] organizes axis in a cylindrical way and each point's value on the star coordinates is simply the projection value onto each dimension. It is good since it uses a single visual cue, uses minimal visual representation and supports interaction. *Heat map* [48] uses different color to represent different values in each cell. *Pixel bar chart* [45] uses pixels within a traditional bar chart to represent detailed information.

Most of the above mentioned visualizations transform the data non-linearly and the actual shapes or trends of the data are not preserved. We therefore lay the foundation of our work mainly on the linear one - the *scatterplot*[121]. It is a projection of the data into an orthogonal 2D basis. In this projection, clusters and their shapes are relatively easy to see, as well as correlations [57]. This is the case when the data dimensionality is low – no occluding could possibly be introduced. But when the number of dimensions reaches a certain large number, points that are distant in high-dimensional space may project into similar locations and this can lead to ambiguities. One possible solution to this is to use the *Scatterplot Matrix* (SPLOM [32]). It compares scatterplots constructed with different bases and arrange them into a matrix. SPLOM is good in the sense that it helps discovering patterns existing in pairwise data dimensional projections, but it would fail to spot any non-axis-aligned pattern and when the dimensionality is very high, integrate information from such a mosaic of plots would be impossible.

Another solution is to use some interface to adjust the compositions of the projection axis and allow it to display projections on arbitrary hyperplane. Biplot is such an example. It visualizes the data dimensions [53] along with the data points in a scatterplot styled display. The problem with BiPlot is that the dimension vectors are overlapping with the actual data and some information can be hidden. If this interface also allows users to dynamically change the projection axis [69][94], it will essentially help users to traverse in the high dimensional space. We will elaborate on this method later in section 2.2.

The various paradigms can also be integrated. Schmid and Interberger [78] combine scatterplot matrices, parallel coordinates, and other displays together. Wong et al. [1] combine and link parallel coordinates with scatterplots matrices, and Yuan et al. [100] integrate parallel coordinates with scatterplots, using MDS to convert multiple axes into a single subplot.

So far, the visualizations we talked about are only using 2D rendering techniques. This is understandable because we are dealing mostly with non-spatial data and a third dimension does not come with it. A possible historical reason for the rare use of 3D techniques in information visualization is that 3D rendering techniques were not well developed, if existing at all, when information visualization came into existence. Already back in the 10th century, people started to use curved lines to trace planetary trajectories [26] and the modern concepts of information visualization (e.g. pie chart, bar chart and histogram) were also already invented in the early 18th century. There are a few exceptions such as 3D scatterplots, cone tree [75] and cushion tree map [97]. But to our knowledge, there hasn't been any information visualization paradigms that fully use the 3D graphic rendering techniques.

When it comes to scatterplot, our work [92] pointed out that the third dimension can add a significant amount of information to the visualization of high-dimensional data. Adding 3D information is essentially about adding depth cues to the scene. This dimension can also fully utilize the innate ability of humans to reason with 3D objects. Adding 3D information is essentially about adding depth cues to the scene.

A relatively small number of depth cues are binocular [96] and they capitalize on the convergence of our two eyeballs. Binocular depths cues are most effective for close-up vision, such as for threading a needle [96]. For objects further away, monocular depth cues are more relevant. In visualization the reliance is mostly on monocular depth cues since binocular depth cues require stereoscopic equipment such as 3D glasses or virtual reality equipment which is not always readily available.

There are many monocular depth cues, such as linear perceptive, occlusion, relative size, accommodation, atmospheric, and shading. Linear perceptive makes use of the sizes or textures of objects to reveal their relative distances to each other. Occlusion exploits the overlap between objects or the shadow created by the light being blocked. Atmospheric cues add imaginary fog into the scene, while accommodation cues render objects in the back less sharp or change the clarity of objects by evaluating if they are in the viewer's focus or not. The shading cue capitalizes on the fact that the visual system typically assumes that lighting comes from above, and the amount of light reflected at surfaces towards the eyes determines the perceived shape and depth of the geometric surface detail [58]. This process is called shading and can range from simple diffuse and specular shading to global illumination [81].

2.2 DISAMBIGUATE HIGH DIMENSIONAL SCATTERPLOT

In order to reason with the visualization, the visualization itself has to provide clear and meaningful information. As is discussed in section 2.1, when visualizing high dimensional data, ambiguities often exist. Numerous research effort has been spent on how to disambiguate high dimensional visualizations – in our case, scatterplots. Those can be divided into two directions – the automatic methods and the manual approaches.

2.2.1 Automatic Methods

The simplest automatic method to provide the users with a set of scatterplots is the Randomized Projection [6]. It has been demonstrated as a simple and computationally efficient method to gain a good overview of a high-dimensional space. It has been successfully applied in text, image and audio data processing [70][12][54] as well as in visual analytics [6]. It produces truly arbitrary views which may contain a varied amount of structures, and are ready to be examined later.

There are other automatic layout optimization approaches seek to generate a view that best fits certain criteria. Given the many possible scatterplot projections it is helpful to design some quality criteria by which to select the most informative views. Research in that area has mainly addressed the selection of axis-aligned views in the presence of clustered or classified data. Sips et al. [82] define a class consistency measure that favors views based on the distance to the class center of gravity or on the entropies of the spatial distributions. Tatu et al. [86] assess quality by measures on density, histogram, and class separation. The rank-by-feature system by Seo and Shneiderman [79] allows users to specify certain statistical criteria, such as correlation, scatterplot uniformity, etc. Schäfer, et al. [77] describe a quality metric that focuses on structural preservation and visual clutter avoidance. GGobi uses projection pursuit [27] to generate interesting multivariate projections [55]. We use a popular evolutionary algorithm – the ant-colony algorithm [19] – in conjunction with view quality metrics such as stress, class density, class separation, holes, and central mass. SeeDB [90] uses a subset of the data and recommend the user with some views the system deems meaning.

A problem with having many projections is also how to manage and organize them. Several map-based diagrams have been proposed in the past [69][102]. We add to this research by presenting a map that is dedicated to the management of generalized subspaces in our Subspace Voyager [94].

Layout optimization schemes, such as Multidimensional Scaling (MDS) [50], Linear Discriminant Analysis (LDA) [64], and t-SNE [61] can help overcome the ambiguity problems of projective scatterplot displays. All have found ubiquitous use in visualization (e.g. [15][37][71]). MDS, for example, seeks to generate a layout where the pairwise distances of points in 2D are relatively similar to those in high-D space.

2.2.2 Manual Approaches

Whatever the objective function and optimization method is, trying to warp high-dimensional space onto a 2D plane is inherently ill-posed since it cannot fully capture multivariate data variations. Distortions are the consequence, which can lead to problems with correctly

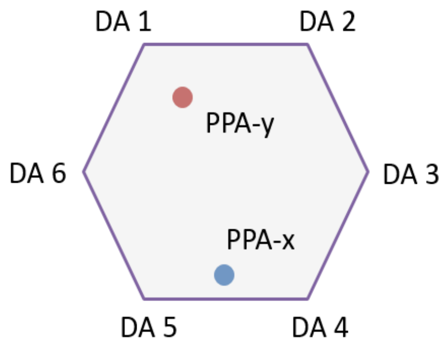


Fig. 1. Pad-based navigation interface. In this case, the PPA-x vector is dominantly a combination of dimension axis DA 5 and DA 6, while PPA-y is dominantly a combination of DA 6, DA 1, and DA 2.

recognizing the true shape and appearance of clusters, and assessing point-wise distances, far and near. Therefore manual interaction is highly recommended in high dimensional data analysis.

This idea of interaction is not new. In star coordinates where data points or clusters that may originate from distant N-D locations may still be mapped to the same 2D area due to their similar dimension vector sums. Hence, users can manually rotate and scale data axes to isolate these points. Some other methods have exploited the concept of 'dynamic transition'. ScatterDice [21] restricts the transitions to motions between two SPLOM tiles, giving rise to a dynamic 3D point cloud projection display. The popular GGobi system [85] is derived from the seminal concept of the 'Grand Tour' [8], also employs trackball controls to smoothly transit between different subspaces. They all enable users to change the projection basis in a continuous fashion, effectively using motion parallax to resolve depth and relative distance. In the realm of health informatics [104], has successfully used different interactions to help the doctors quickly understand patient history and make the diagnose. In sport analysis, Shao et al. [80] has used visual interactive search to analyze soccer trajectories.

A recently work by J. Nam and K. Mueller [69] presents the system of 'dynamic scatterplot'. It has fully embraced this paradigm, enabling projections from any orientation. Its interface is sketched in Fig. 1. This navigation pad consists of a polygon with *S* vertices, where *S* is the cardinality of the subspace. Each vertex corresponds to a native dimension and so the subspaces are axis-aligned (and not generalized). It should also be noted that for S>3 different orderings of the vertices are required to allow users to access the full projection coverage of the subspace.

The interior of the polygon shows two disk-shaped pointers. They represent the two (N-D) basis vectors into which the N-D point cloud is projected for display using the vector dot product. In [69] these two vectors are called *Projection Plane Axis (PPA) vectors* – the x-axis is PPA-x and the y-axis is PPA-y. The vectors are computed from their position in the pad polygon via generalized Barycentric coordinate interpolation [65].

In the pad-based interface users can control the influence a dimension has on the display by moving either the PPA-x or PPA-y pointer towards that dimension. This essentially spreads out the projected point cloud along that dimension and so reveals the dimension's ability to separate the data points into different populations/clusters. Then, by moving the other pointer towards another dimension bivariate relationships can be visualized. Finally, when moving either or both pointers midway toward a set of dimensions users can appreciate their combined effects due to multivariate relationships.

While this pad-interface allows unprecedented control in the dynamic manipulation of the view onto the N-D point cloud the need to separately manipulate two pointers in sequence suffers from a certain lack of ergonomics. A further shortcoming is that users is required to keep track of two interfaces at the same time: (1) the visualization window that shows the moving point cloud along with a projected coordinate system, and (2) the pad that controls the orientation of the projection plane. In practice, a user may observe one or more dimensions that should be emphasized in the display as they might offer the potential to break up a cluster into two or more components. In that case this user would need to first check the pad in what direction a pointer should be moved and also which one, and then perform the move. In the present work, we aimed for an interface that makes this operation more straightforward by embedding the navigation controls directly into the display. Enhancing the well-known trackball interface with N-D navigation capabilities seemed to be good choice towards this goal.

We should note that the automatic and manual approaches can be integrated. For example, Albuquerque et al. [3] combines quality metrics with the brushing technique to analyze high dimensional data. Our work [94] also uses the Ant Colony Optimization [19] algorithm powered projection pursuit to optimize views along with various interactions to let users explore the data.

2.2.3 Subspace Analysis

Both the automatic and manual methods we talked about are indeed endeavors to find meaningful subspaces in the high dimensional space where structures can be revealed. Subspace clustering has been an active research area in the data mining community [49] but the focus was mostly on automated algorithms. In the field of visualization, one may distinguish the contributions by how much they rely on automated subspace analysis methods. On one end are the works by Yuan et al. [101] and Kim et al. [47]. The former proposes a visual subspace exploration approach that focuses mainly on interactive dimension set selection and refinement and the latter proposes a system where users can drop data points into two different groups and the PPA axes would be updated automatically. On the other end are the approaches that first perform an automated subspace clustering step and then visualize the results either as small multiples of scatterplot projections, as MDS layouts, or view the dimensions that define the subspaces [9][87]. Some other approaches [59] also used animation to traverse between the generated subspaces to see how those are related. We also first perform clustering but then use the results to provide guidance in the subsequent visual exploration of the actual subspaces, focusing on cluster appearance and relations which can be helpful in the visual reasoning process.

2.3 GENERATE HIGH DIMENSIONAL DATA

Synthetic data is useful in many fields. There are software for users to edit existing data, such as OpenRefine [122] or the visual framework Baudel [10] provides to edit large datasets. But using those tools it is not possible to produce a new dataset from scratch. To produce brand new data, if users are familiar with the setting, traditional practice is to run a script specifying certain criteria, otherwise, statistical software or environment such as R or Matlab could be used. Some data generation algorithms have also been proposed, they can be based on constraints [18] [29], genetic algorithms [72] [2] [3], optimization and search [62][89][67] or chaining approach [23].

When it comes to sophisticated data generation software, [5] proposes a way to generate high dimensional data by inserting points in 2D projection and back projecting it to the original data space. This method sounds plausible until users embed those points back to 2D projection – a significant distortion could be noticed. The reason is when performing back projection, only locally linear embedding is guaranteed and it’s therefore not WYSIWYG. Albuquerque et al. [4] recently presented a system for high-dimensional dataset generation. It is based on drawing and is interactive, but it can only generate up to 2.5D dataset and when applying this tool in the high dimensional space, its limitations cannot be neglected. But to our knowledge, there is no specific tool – let alone sketching based tool – for complex high dimensional data generation.

CHAPTER 3

STUDY ON INTRINSIC DIMENSIONALITY

In order to quantitatively test if adding a third dimension in high dimensional data visualization is helpful, we studied a variety of real-world datasets and confirmed that the extension from the traditional 2D space to 3D space is indeed justified – most datasets we studied had clusters residing in subspaces with more than two significant principal components. This findings gave us the motivation to continue on the path of decomposing high dimensional space into a series of 3D subspaces and analyze them.

3.1 MOTIVATION

Does 3D really make sense for data visualization? Or, more specifically, does it make sense for visual cluster analysis, which is a sub-field of visualization? This principal question might be framed in visual perception theory. As is discussed in the introduction, we human beings live in a 3D world and our neural network is trained to observe, explore, analyze and reason with 3D objects and relations. Scientific visualization has long incorporated all three dimensions but for information visualization where non spatial data are being visualized, 3D methods are not wildly applied. The question is, would it be redundant to build 3D interactive analyzing system? Or current 2D visualization frameworks are enough? To answer this question, we designed a series of studies on the *Intrinsic Dimensionality* (ID).

In this current research, we were particularly interested in finding out how appropriate the simple 3D display would be in practice. For this purpose we studied a variety of representative datasets to determine the ID characteristics of the subspaces in each. Subspace clustering can significantly reduce the ID of the data. It essentially decomposes the high-dimensional data into a composite of lower-dimensional independent phenomena, for which a 3D display or a transitional display that is not overly more complex might be sufficient. To find these subspaces we used the well-established DBSCAN clustering algorithm [22], augmented by a visual interface that gave us some insight into proper parameter settings to reach the different clusterings quickly.

3.2 METHOD

We do not assume any prior classification of the data. We consider each cluster a sub-space of the data. Here, a cluster is a set of ε-connected points where ε is the minimal distance a point must have to some other point in the cluster to also be part of this cluster. This property is not fulfilled by the k-means algorithm but is common in sub-space clustering. The outcome of this

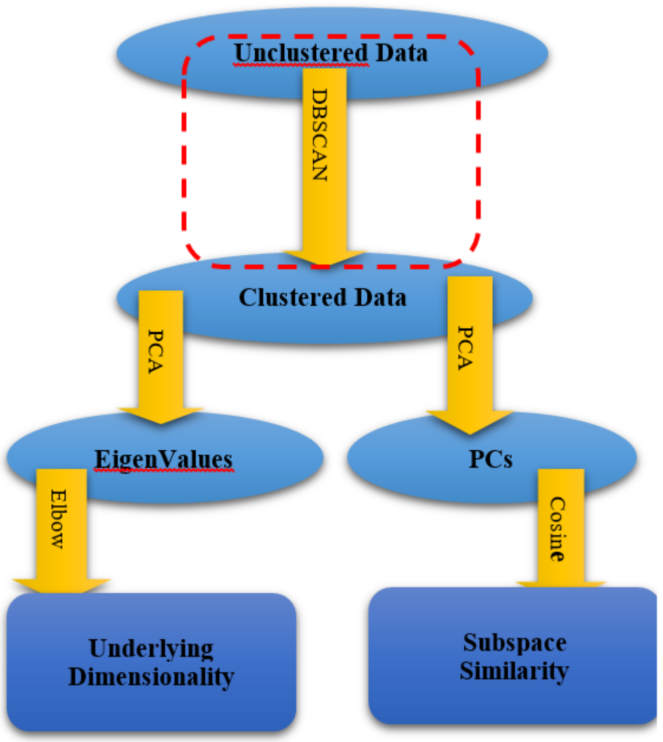


Fig. 2. Study workflow. We firstly performed density-based spatial clustering via DBSCAN to obtain a set of clusters of arbitrary shape. Next, we did PCA on each cluster and used the elbow method/scree plot to estimate their intrinsic dimensionality. We also compared the PCs of different clusters via the cosine similarity to determine if the clusters exist in the same or different subspaces.

clustering is a set of sub-spaces and associated shapes, each of which has a certain intrinsic dimensionality (ID). The ID determines the complexity of the shape display needed to visualize it.

A classic method to discover the ID is Principal Component Analysis (PCA). If ID=2 then a conventional scatterplot will do. On the other hand, if ID=3 then a simple 3D display is sufficient. And finally, if ID>3 we need to allow users to transition between multiple 3D shapes, one for each distinct PCA vector set of three.

To determine whether a 3D display would suffice for visual cluster analysis, we conducted a series of studies on a variety of unclustered datasets, ignoring any classification when available. The workflow of our analysis is depicted in Fig. 2. As a first step, for each dataset, we performed density-based spatial clustering via DBSCAN to obtain a set of clusters of arbitrary shape. Here, the tuning of the DBSCAN parameters can give rise to different numbers of clusters. Next, we ran Principal Component Analysis (PCA) on each cluster and used the elbow method/scree plot to estimate their intrinsic dimensionality. We also compared the PCs of different clusters via the cosine similarity to determine if the clusters exist in the same or different subspaces. In the following sections we describe our study methodology in detail and discuss its results.

3.2.1 DBSCAN

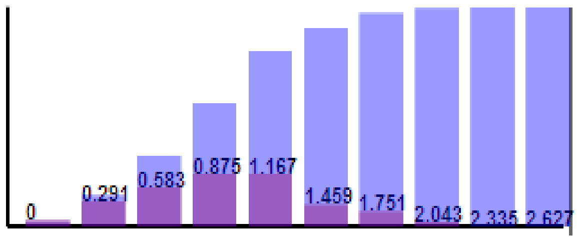
DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise and is one of the most widely used and cited data clustering algorithms. Its key concept is to define clusters based on the notion of *reachability*. Gven two points, p and q, if the distance between them is less than ε and q has a sufficient number of neighbors within the ε distance, we say p is *directly density-reachable* from q. On the other hand, p and q are *density-reachable* if there exists a sequence of points p_1, p_2, p_3 … p_n where p_{k+1} is directly density-reachable from p_k (k = 1,2,3…n-1), then p_1 = p and p_n = q. Finally, if there is a third point r from which both p and q are density-reachable, p and q would be *density-connected*. Every point-pair inside one cluster found by DBSCAN must be density-connected, and if a point is density-reachable from any point within one cluster, it also belongs to that cluster.

DBSCAN requires two parameters: the neighborhood radius ε and the minimum number of points (*minPtn*) that a cluster should at least have. It also uses a flag to distinguish whether a point has already been processed or not. DBSCAN starts with an arbitrary yet unvisited point and finds all points that are no further than ε to it. If this number of points is greater than *minPtn,* a new cluster is started else the point is classified as noise. If a cluster is formed, all discovered points are added to the starting point's neighbor list. Next, for every point in the list (note that the elements of the list are dynamically added), its ε-neighborhood is also retrieved. If it is also dense (the number of points being larger than *minPtn*), all of its ε neighbors are also added to the list. This process continues until no density-connected points can be further discovered. Then DBSCAN finds the next unvisited point and repeats this process.

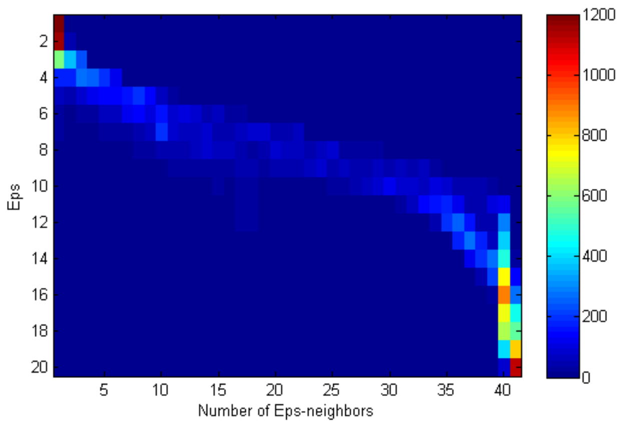
DBSCAN is not parameter free – it requires users to choose the proper combination of ε and *minPtn*. DBSCAN is quite sensitive to these two parameters, but there is no general guideline on how to set them. And so, finding the settings that resulted in a defined change typically required much time consuming trial and error.

A first solution could be to run all possible setting as a background process and survey the clusterings that result. But this can take a considerable amount of time for reasonably sized datasets. Instead, we designed two visualizations that convey some idea about the relationships in the data and so provide some assistance in choosing the parameters

The first of these visualizations is a distance histogram (Fig. 3(a)) which shows all pairwise distances between points. The purple bars are the normal histogram while the blue bars are the cumulative histogram which shows the setting at which the sharpest changes occur. These histograms convey the distance distribution of the data and allow users to pick specific ε-values that will likely give rise to a change in the clustering.



(a) Distance histogram. This shows all pairwise distances between points. The purple bars are the normal histogram while the blue bars are the cumulative histogram which shows the setting at which the sharpest changes occur.



(b). Heat map describing the number of points that have a certain number of ε-neighbors. We constructed this plot by visiting each point, counting the number of neighbors for each discretized ε-setting, and incrementing the corresponding 'number of neighbors' bin of the plot.

Fig. 3. . DBSCAN visualizations

The second visualization is a 2D heat map (Fig. 3(b)) that visualizes the pairwise distances over the number of neighbors that are within each such distance. We constructed this plot by visiting each point, counting the number of neighbors for each discretized ε-setting, and incrementing the corresponding 'number of neighbors' bin of the plot. The plot allows users to estimate how many points would have a certain number of neighbors residing within a certain ε-distance, which can be helpful when choosing *minPtn* (and ε). In this particular example, we learn that the relationship is a fairly narrow curve, and so this plot saves users the considerable amount of time trying out *minPtn*-ε combinations that fall into the vast blue areas of the plot.

3.2.2 Intrinsic Dimensionality Analysis

Following DBSCAN, we perform PCA on each discovered cluster. PCA uses an orthogonal transformation to find linear uncorrelated variables (the PCs) that describe the data. The strength of each PC vector – the eigenvalue – determines the amount of variation in the data it can

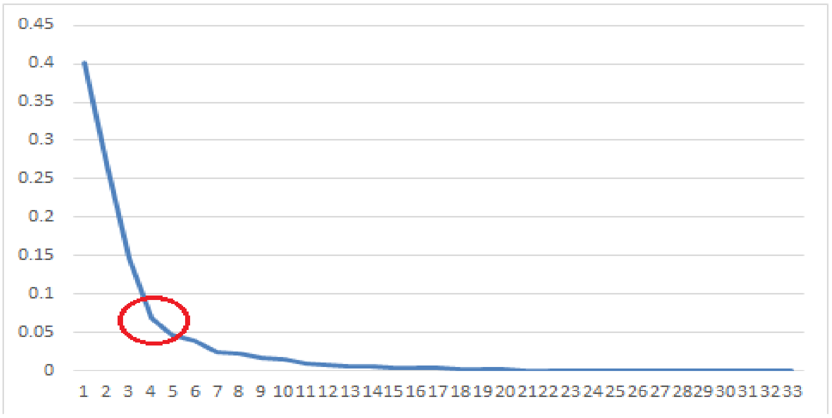


Fig. 4. Elbow method. The intrinsic dimensionality can be estimated by locating the scree plot's elbow– the point on the scree plot curve at which it stops to decrease significantly.

explain. Normalizing these eigenvalues by the overall sum of eigenvalues expresses this strength in percent.

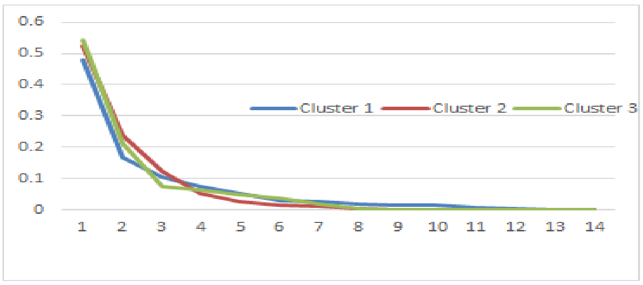
After obtaining these normalized eigenvalues and discarding those with values less than 0.001 we create a *scree plot* – an ordering of the eigenvalues from largest to smallest. The intrinsic dimensionality can then be estimated by locating the scree plot's *elbow* or *knee* – the point on the scree plot curve at which it stops to decrease significantly [88]. A simple metric to find this elbow is to draw a line from the first to the last point of the curve and then find the point that is farthest away from that line (see red circle in Fig. 4).

While the elbow criterion provides a clear and deterministic way to decide the intrinsic dimensionality, we (and others [91]) found that often the elbow is not overly well expressed. The curve only slowly bends and a slight variation of the elbow metric can change its location drastically. Instead, it might be more appropriate to also look at the percent contribution of the eigenvalues. While we have not formally tested this, a contribution below a significance value of 0.05 (5%) may not account for much variation in the visual projection display. For the cluster plotted in Fig. 4this would then point to an intrinsic dimensionality of 4-5.

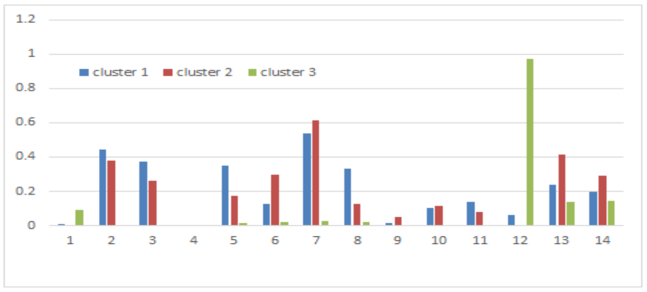
Finally, we also conduct a similarity analysis of the subspaces found for a given dataset. We compute the cosine similarity for each significant PC pair for the two associated subspaces, and their normalized sum indicates if the two clusters reside in the same or similar subspace, or far apart.

3.3 CASE STUDIES

We studied a variety of datasets, most from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/). Below we present three representative results from this study. For



(a) Scree plot



(b) Value histogram

	Similarity (cluster 1 & 2)	Similarity (cluster 1 & 3)	Similarity (cluster 2 & 3)
PC1	0.91	0.15	0.13
PC2	0.75	0.48	0.67
PC3	0.56	0.38	0.49
PC4	0.82	0.52	0.52

(c) Table: similarity between PCs

Fig. 5. Boston housing dataset. The scree plot in (a) shows the intrinsic dimensionality for all three clusters. Only cluster 3 has a clear elbow at PC=3. Cluster 1 has it there as well, while cluster 2 has it at PC=4. Hence, a 3D display will be appropriate for all subspaces. (b) is the value histogram for PC1 and the table in (c) presents the cosine similarities between pairwise PCs. We notice PC1 for cluster 3 is quite different from the PC1 of the other two clusters and the remaining PCs also only have a similarity of about 0.5 with those of cluster 1 and 2. We hence conclude that (1) cluster 3 resides in a rather different subspace than cluster 1 and 2, and (2) the subspaces of cluster 1 and 2 are somewhat closer.

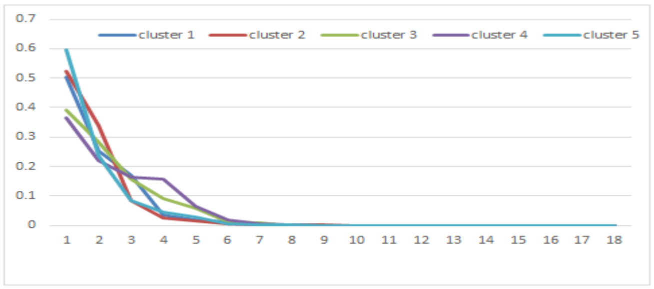
each dataset, we first decide the intrinsic dimensionality of the clusters using the scree plot and then compute their cosine similarity.

3.3.1 Boston Housing Data

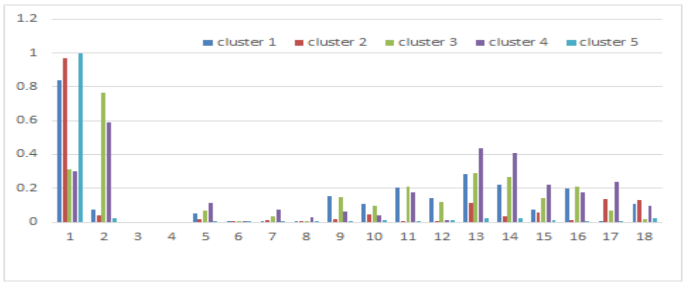
This dataset [123]describes housing values in suburbs of Boston. It has 506 instances with 14 continuous attributes. We set ε equal 0.5828 and *minPtn* to be 12. After running DBSCAN, we obtained three clusters. Fig. 5(a) shows the corresponding scree plot.

We observe that for all three clusters the amount of variance covered by the third PC dimension is at or above 5%, and it is at or above 10% for clusters 1 and 2. Only cluster 3 has a clear elbow at PC=3. Cluster 1 has it there as well, while cluster 2 has it at PC=4. Hence, a 3D display will be appropriate for all subspaces.

Fig. 5(b) shows the value histogram for PC1 and the table in Fig. 5(c) presents the cosine similarities between pairwise PCs. We observe that for clusters 1 and 2 the similarity of their most significant PC, PC1, is 0.91, while for the remaining three PCs, the similarity drops only slightly to 0.75, 0.56 and 0.82, respectively. On the other hand, PC1 for cluster 3 is quite



(a) Scree plot



(b) Value histogram

Similarity / PC	Cluster 1 & 2	Cluster 1 & 3	Cluster 1 & 4	Cluster 1 & 5	Cluster 2 & 3	Cluster 2 & 4	Cluster 2 & 5	Cluster 3 & 4	Cluster 3 & 5	Cluster 4 & 5
PC1	0.89	0.60	0.62	0.86	0.40	0.44	0.97	0.92	0.34	0.24
PC2	0.33	0.37	0.69	0.70	0.33	0.36	0.59	0.45	0.29	0.90
PC2	0.79	0.48	0.82	0.34	0.67	0.57	0.57	0.52	0.62	0.31
PC4	0.32	0.63	0.65	0.56	0.56	0.74	0.55	0.74	0.56	0.52

(c). Table: Similarity between PCs

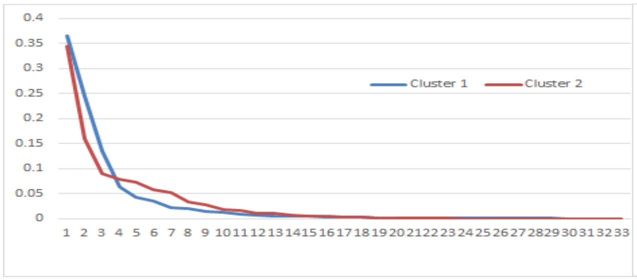
Fig. 6. Image segmentation dataset. From the scree plot in (a), we find that for all clusters the amount of variance covered by the third PC vector is in the range of 10-15%, the elbow is at the 4th eigenvalue and the 5% significance is reached at the 4th and 5th. So a 3D display with transitioning capabilities will be helpful. (b) shows the value histogram for PC1 and (c) presents the cosine similarities between pairwise PCs. Here we observe that probably the most similar clusters are cluster 1 and 5 as they have the most consistent PC vector similarities. Other clusters seem quite disparate.

different from the PC1 of the other two clusters (0.15 and 0.13, respectively), and the remaining PCs also only have a similarity of about 0.5 with those of cluster 1 and 2. We hence conclude that (1) cluster 3 resides in a rather different subspace than cluster 1 and 2, and (2) the subspaces of cluster 1 and 2 are somewhat closer.

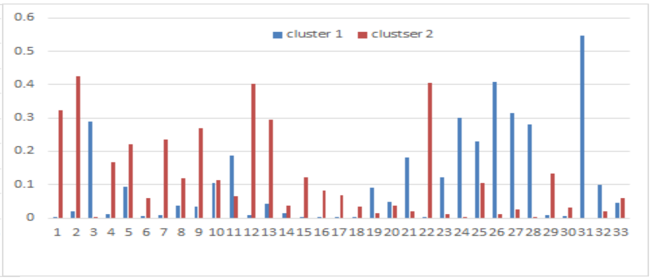
3.3.2 Image Segmentation Data

This dataset [117] is composed of feature vectors derived from 1,200 3×3 image patches – 300 random instances each from four image classes (Brickface, Cement, Foliage, and Grass). The feature vectors have 19 attributes (dimensions) which are statistical measures of the images, such as region centroid, region pixel count, density, hue, and others. The third attribute 'region-pixel-count' is 9 for all instances. We removed it and are left with 18 all-numerical attributes. We set ε=0.321 and *minPtn*=47 and obtained five clusters.

From the scree plot shown in Fig. 6(a) we find that for all clusters the amount of variance covered by the third PC vector is in the range of 10-15%, the elbow is at the 4th eigenvalue and the 5% significance is reached at the 4th and 5th. So again, a 3D display with transitioning



(a) Scree plot



(b) Value histogram

	Similarity(cluster 1 & 2)
PC1	0.143
PC2	0.105
PC3	0.276
PC4	0.305

(c) Table: Similarity between PCs.

Fig. 7. ISDAC dataset. We notice that at least three PCs are required to capture 90% of the data variance. The PCs for the two clusters are quite different (low cosine similarity values and very different PC1 value histograms) and hence the two clusters belong to entirely different subspaces.

capabilities will be helpful – a single 2D projection will not be able to capture the variances sufficiently.

Fig. 6(b) shows the value histogram for PC1 and Fig. 6(c) presents the cosine similarities between pairwise PCs. Here we observe that probably the most similar clusters are cluster 1 and 5 as they have the most consistent PC vector similarities. Other clusters seem quite disparate.

3.3.3 ISDAC Data

The ISDAC dataset [103] is an atmospheric dataset fused from multiple sources and consists of 221 data points, each a 33-dimensional vector composed of latitude, longitude, altitude, time stamp, temperature, and pressure and measurements on the cloud particles (cloud droplets presence, cloud particle concentration, etc.) and on aerosol particles (size and composition: soot, sulfate levels, organics, dust, sea salt, etc.). We set ε=0.7592 and *minPtn*=6 and obtained two clusters. Fig. 7 shows the results we obtained. We notice that at least three PCs are required to capture 90% of the data variance. The PCs for the two clusters are quite different (low cosine similarity values and very different PC1 value histograms) and hence the two clusters belong to entirely different subspaces.

CHAPTER 4

THE SUBSPACE VOYAGER FRAMEWORK

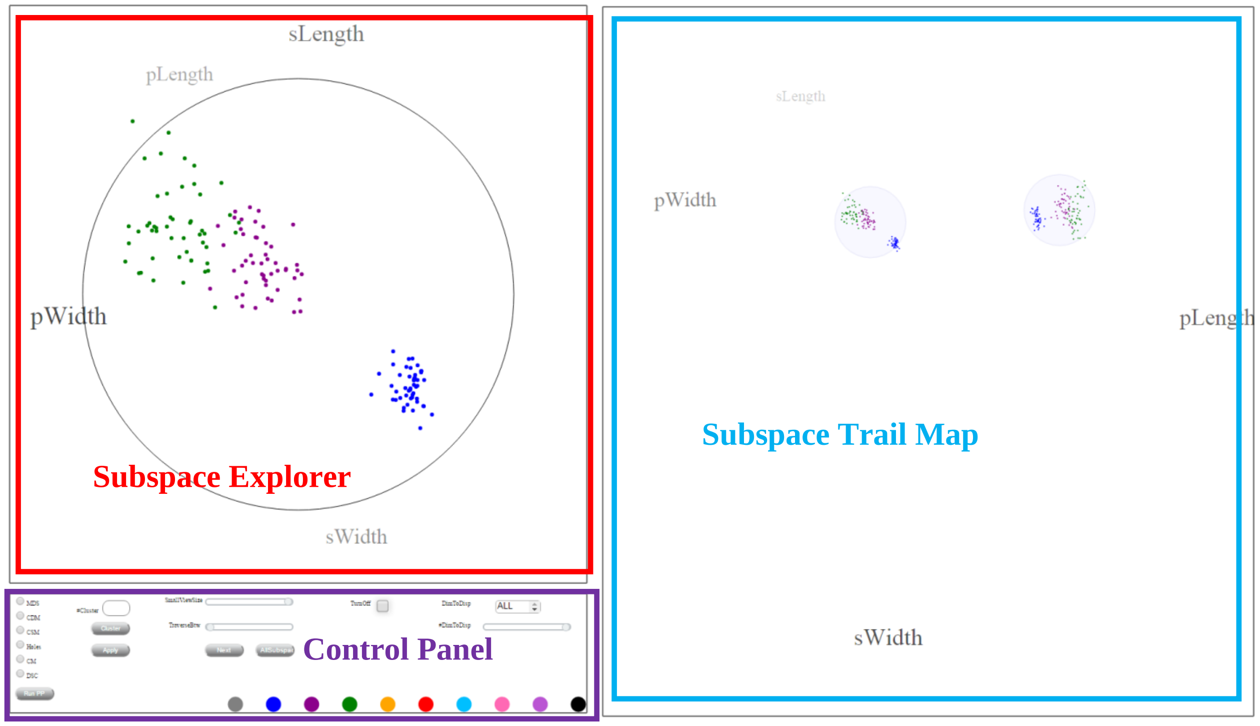


Fig. 8. Subspace Voyager interface. It has three main components: the Subspace Explorer (SE), the Subspace Trail Map (STM) and the control panel. The SE is coupled with the trackball interface. It not only displays the data as a scatterplot, but it also allows users to visualize the current directions of the projected dimension axis vectors as labels placed outside its circular boundary. SE offers various interactions for users the exam the data. The STM holds a set of views (and their parameters) that users may have found interesting during the exploration by embedding them in to a word cloud of attributes. The control panel allows user to set the various parameters and modes in the system

Analyzing high-dimensional data and finding hidden patterns is a difficult problem and has attracted numerous research efforts. Automated methods can be useful to some extent but bringing the data analyst into the loop via interactive visual tools can help the discovery process tremendously. An inherent problem in this effort is that humans lack the mental capacity to truly understand spaces exceeding three spatial dimensions. To keep within this limitation, we describe a framework that decomposes a high-dimensional data space into a continuum of generalized 3D subspaces. Analysts can then explore these 3D subspaces individually via the familiar trackball interface, but using additional facilities to smoothly transition to adjacent subspaces for expanded space comprehension. Since the number of such subspaces suffers from combinatorial explosion, we provide a set of data-driven subspace selection and navigation tools which can guide users to interesting subspaces and views. A subspace trail map allows users to manage the explored subspaces, and keep their bearings and return to interesting subspaces and views. Both trackball and trail map are each embedded into a word cloud of attribute labels. We demonstrate our system via several use cases in a diverse set of application areas, such as cluster analysis and refinement, information discovery, and supervised training of classifiers. We also conducted a user study to evaluation the usability of the various interactions our system provides.

4.1 MOTIVATION

High-D space is generally confusing to most people since humans do not possess the innate neural network to recognize and reason with high-D objects. Spatial reasoning skills are acquired in early childhood where often haptic and visual experiences are combined to build 3D mental models of the real world. Since high-D objects are largely mathematical and do not occur in a tangible form, the associated cognitive reasoning chains are not developed in these critical early years. This lack of reasoning faculties represents a barrier for most people when dealing with high-D data later in life and so deprives them of the chance to find more insight in these data.

We describe a framework and interface that eases this barrier by design, called the *Subspace Voyager*. It serializes the exploration of high-D space into a continuous travel along a string of generalized 3D subspaces. This serialization allows us to abolish the complex interactions and representations that are often typical to high-D space exploration tools and replace them with paradigms familiar to most people, such as trackballs, maps and word clouds. Our interface uses these to help users explore the generalized 3D subspaces, navigate the continuum of 3D subspaces, and assess the relevance of individual attributes for a given subspace, respectively.

The simplicity gained through the 3D subspace decomposition comes at a price – the extent of the transformations defined on such a restricted subspace is limited and may not reach far enough to generate a projection in which a pattern of current interest is well expressed. To enable a reach beyond these limits we have augmented the trackball with extra capabilities that allow users to "chase" the discovered patterns by moving to adjacent 3D subspaces via simple mouse interactions. In this way patterns can be observed that are truly multivariate and not restricted to a single 3D subspace.

In some sense our approach is akin to that taken in an upcoming Indie video game, *Miegakure* [105] (itself inspired by the classic novel *Flatland [1]*) which enables 4D space travel by swapping one of the three current dimensions. However, we go significantly further than this game – our spaces are much greater than 4D and we also allow transitions in all dimensions simultaneously. Yet, it is encouraging that the entertainment industry sees fun in this type of space travel. It suggests that our interface might be fun and engaging as well, which will immensely benefit the analytics that is performed with it.

The 3D subspaces our system supports are general in the sense that they are formed by an arbitrary orthogonal axis system which are not necessarily axis aligned. As such they allow ample freedom for users to visualize high-D phenomena that are not aligned with a set of data axes. This, however, brings about a huge number of subspaces. To manage this complexity we provide a number of objective-driven search and clustering facilities that assist users in locating subspaces with interesting structures.

When designing our interface we placed great emphasis on making the interactions as direct and intuitive as possible. Most exploration goals can be achieved by expressing them directly in the visualization, via simple mouse selections and transitions. At the same time, our framework is quite general and is readily applicable for many tasks and application areas that involve multivariate data diverse, such as cluster sculpting [68] and analysis, information discovery, and supervised training of classifiers, just to name a few.

The contributions of our work are:

- Propose a BiPlot [53] based display of high dimensional data but avoids the typical overlapping of vectors in BiPlot.
- Implement different ways to interact with the data and to analyze the data in an exploratory fashion.
- Apply Ant Colony Optimization Algorithm [19] to optimize views according to user specific criteria.
- A Principal Component Analysis [40] based map records all important findings.
- A presentation mode to let users present their finds via animation.

4.2 SYSTEM OVERVIEW

Fig. 8 shows the Subspace Voyager interface. It has three main components: the *Subspace Explorer (SE)*, the *Subspace Trail Map (STM)* and the control panel, which allows users to set the various parameters and modes in the system.

The exploration pipeline of the Subspace Voyager is outlined in Fig. 9. After loading the data, our system performs either Random Projection or Subspace Clustering and Principle Component Analysis (PCA) [34] to build the initial 3D subspace. The data is then projected onto this subspace and is displayed in the trackball embedded in the *SE*. There are different interactions users can perform on the trackball. The first one is to rotate the trackball while pressing down the mouse left button. This allows users to explore the current subspace. The second one is to change the weightings of data dimensions quickly by moving the mouse toward their labels displayed along the trackball while pressing down the right mouse button. This lets users jump to different subspaces and explore the data with a higher emphasis on one or more attributes of interest. Users can also tag points by brushing them into different colors. Our system can run Ant Colony

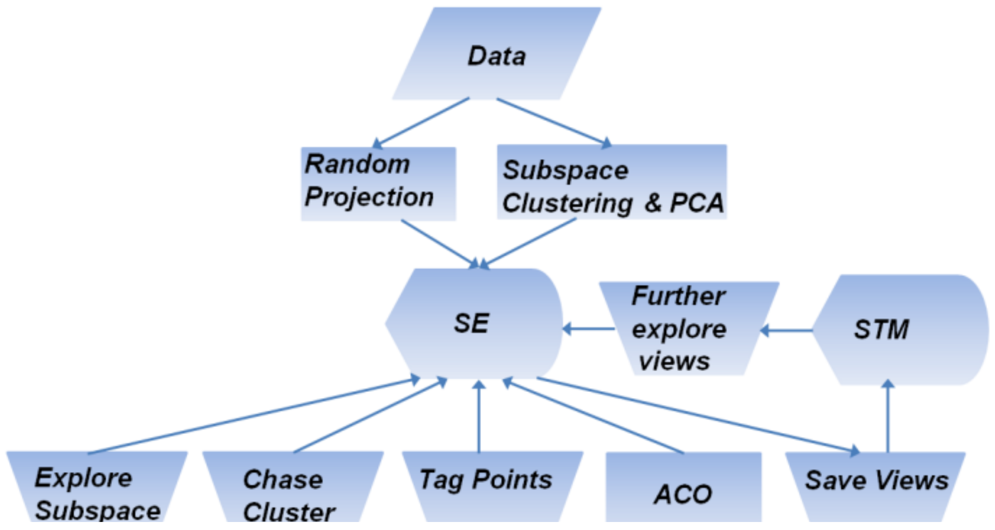


Fig. 9. The workflow of our *Subspace Voyager*. It starts with projecting the input data onto the initial 3D subspace obtained either by Random Projection or Subspace Clustering and Principle Component Analysis (PCA). The projected data is displayed in the trackball in our *Subspace Explorer (SE).* Users can either rotate the trackball and explore the current subspace or change the weightings of data dimensions quickly to jump to different subspaces. Users can easily tag points by brushing them into different colors. Our system can run Ant Colony Optimization Algorithm (ACO) to optimize views according to user selected criteria. At any time, users can save the current view in the trackball to the *Subspace Trail Map (STM)* to keep track of interesting findings. Users can also bring any view in the STM back to the trackball for further exploration

Optimization Algorithm (ACO) [19] to optimize views according to user selected criteria. At any time, users can save the current view in the trackball to the *STM* to keep track of interesting findings. Users can also bring any view in the STM back to the trackball by dragging for further exploration.

Choosing meaningful subspaces for exploration is a key challenge in multivariate data analysis and much work has been dedicated towards this goal, as discussed in Section 2. We have implemented two such strategies: (1) randomization and (2) subspace clustering. On the one hand, our interface enables users to generate R (where R is user selectable) random 3D subspaces which are further optimized using ACO powered projection pursuit (see Section 5.4). We use the technique proposed by Anand et al. [6] to generate these subspaces. We then use view optimization to generate a scatterplot projection of the subspace and insert it into the STM. On the other hand, we also use clustering to generate a set of meaningful and salient subspaces. Here we assume, similar to Liu et al. [59] and also our own work [92], that each cluster forms a subspace on its own. We characterize each such subspace by its three principal components (PCs) obtained via Principal Component Analysis (PCA)[40]. We then optimize a 2D view from this 3D subspace, insert it into the STM.

We should also note that in a view that has the PC vectors as its basis, if two dimension vectors are very close, it means they are to some extent correlated. This is especially true when two dimensions both have large weightings in one significant PC – then these two dimensions are strongly correlated [106]. We will make use of this relationship in the use case described later.

The Subspace Explorer (SE) is coupled with the trackball interface. It not only displays the data as a scatterplot, but it also allows users to visualize the current directions of the projected dimension axis vectors as labels placed outside its circular boundary. This display can be considered as a BiPlot [53] style visualization. Instead of using the length of the vectors to indicate how well the respective attribute is expressed in the current view, the size and opacity of these labels indicates – larger and bolder font means more of the attribute's variability is shown. Conversely, the label placement reveals the radial direction along which the variability is mostly exposed (Fig. 8). The simplest form of trackball interaction generates scatterplot projections confined to the current (generalized) 3D subspace projected into the SE. This projected generalized 3D subspace can be modified by:

- Trackball interaction: users can transition to adjacent 3D subspaces by augmented trackball interaction
- Randomized projections: this discovers new 3D subspaces ready for trackball-based exploration
- 3D Subspace interpolation: moving a slider in the control panel generates intermediate 3D subspaces between two subspaces in the STM that can be explored with the trackball
- View optimization: the 3D subspace (as well as the current projection in the current 3D subspace) can be optimized via projection pursuit driven by a user-defined set of criteria

The control panel provides several options for trackball use. The checkbox 'TurnOff' specifies if all data points are to be shown or only those that are well described in the current subspace, i.e., belong to that subspace. The color bar on the bottom right corner is the brushing tool. It allows users to tag points or groups of points in a dedicated color to cluster them or mark them as inactive in grey.

The Subspace Trail Map (STM) holds a set of views (and their parameters) that users may have found interesting during the exploration by embedding them in to a word cloud of attributes. We treat each view as one point and use PCA on all of them to spread them out. In case of overlapping, the 'SmallViewSize' slider can be used to change the size of the small views. Users can bring any view back to the trackball for further exploration. They can also use the STM as a media to present important findings to audience via animation.

4.3 THE SUBSPACE EXPLORER AND TRACKBALL INTERFACE

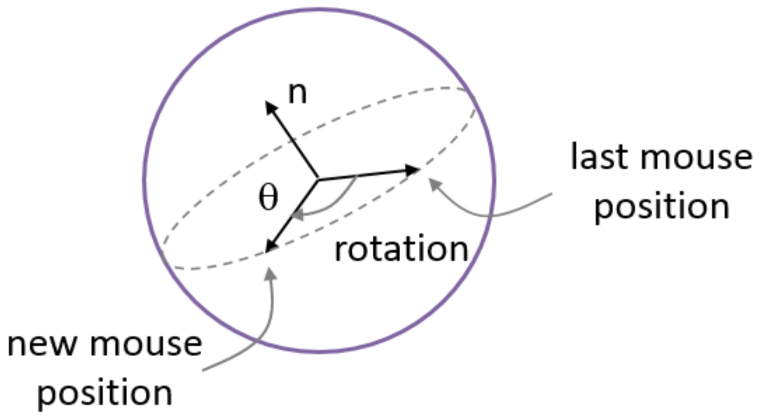


Fig. 10 3D trackball concept

Users can tilt the trackball and watch the resulting scatterplot react to the motion. Fig. 10 sketches how a trackball works. Imagine a virtual sphere that encapsulates the current generalized 3D subspace. When clicked, the screen coordinate of the mouse is mapped to this sphere. Given the current and previous mouse clicks, we can compute the axis of rotation *n* and the rotation angleθ. From those two quantities a 3×3 rotation matrix as derived, as described in [7].

4.3.1 CREATING THE TRACKBALL SPACE PROJECTION MATRIX

The trackball system only works in 3D but our data points are N-D and so we need to project the ND points into 3D before rotating. We achieve this by post-multiplying the trackball rotation matrix T with the 3×N projection matrix P. We have two options for the first two of the vectors in P: (1) the orthogonal PPA x-axis and y-axis pair we have obtained from the randomized projection procedure, or (2) the two most significant PCs we have obtained when performing PCA for the selected cluster. In both cases we require a third orthogonal axis, call it the PPA z-axis. Since this is N-D space we have a great number of choices here. To select an initial vector we can:

- Randomly generate an N-D vector
- If the PPA x-axis and PPA y-axis are generated via PCA, use the third most significant axis for the PPA z-axis

Note that the resulting vector is not necessarily orthogonal to the PPA x-axis and the PPA y-axis. To make it orthogonal we use the Gram-Schmidt orthonormalization process [31]. The Gram-Schmidt process takes N linearly independent vectors and produces N orthonormal vectors spanning the same N-D space. Let $<x_1, x_2>$ denote the inner product of two vectors x_1 and x_2

and let ||y|| denote the length of vector y. Let the projection of vector x_2 onto vector x_1 be $proj_{x_1}(x_2) = x_1 \frac{<x_1,x_2>}{<x_1,x_1>}$. Gram-Schmidt starts with N linearly independent N-D vectors $\{x_1, x_2, x_3, \dots, x_N\}$ and performs the following calculations:

$$y_1 = x_1, \qquad e_1 = y_1/\|y_1\|$$

$$y_2 = x_2 - proj_{y_1}(x_2), \qquad e_2 = y_2/\|y_2\|$$

$$y_3 = x_3 - proj_{y_1}(x_3) - proj_{y_2}(x_3), \qquad e_3 = y_3/\|y_3\|$$

$$\vdots$$

$$y_N = x_N - \sum_{j=1}^{N-1} proj_{y_j}(x_k), \qquad e_N = y_N/\|y_N\|$$

The resulting vector set $\{e_1, e_2, e_3 \dots, e_N\}$ is the desired orthonormal set. In practice we keep the PPA x-axis and PPA y-axis which are already orthonormal and run Gram-Schmidt to orthonormalize the PPA z-axis from the initially chosen vector. Once P is configured in this way, T is reset to the identity matrix, ready to be manipulated in the 3D trackball interaction.

4.3.2 Processing the Points within the Trackball Space

With P in place, the following sequence of operations is executed for every trackball move: (1) compute the 3×N compound projection matrix M=S·T·P, where S is an optional scaling matrix that allows zooming into the display, and (2) multiply each N-D point vector VND by M to get the 3D points V3D=M·VND. But ultimately we are only interested in the projection of the points into the coordinate system spanned by the PPA-x and PPA-y vectors manipulated which the trackball. This yields a set of 2D points V^{2D} which are the first two components of V^{3D} since the projection is orthogonal.

We have not observed a significant delay in the direct projection of N-D points in the operation of the trackball. But first pre-computing a 3D point cloud right after construction of the 3D coordinate system and rotating them directly for the lifetime of P can reduce the number of computations to roughly N/3 of the original computations. We have not chosen this intermediate step because: (1) it requires extra storage which can be significant for large point clouds, and (2) it incurs some delay in the initial response of the trackball whenever a new P is created. Given the capabilities our system provides, this can occur quite frequently (see section below).

4.3.3 Mouse Interactions within the Trackball Interface

We provide three different kinds of mouse interactions, all controlled with different mouse buttons pressed in our trackball interface.

The first one is to explore a generalized 3D subspace. This is the basic mouse interaction performed when the left button is depressed. The kind of operation we perform here is exactly the same as described in Sections 3.3.1 and 3.3.2. Referring to the pad-based interface [69], it corresponds to a triangular pad-polygon but now the vertices are not due to original dimension axes but to arbitrary vectors. This greatly generalizes the adaptability of the display to the orientation of high-D features which might have required many more original dimensions to cover. It but effectively avoids the need for a pad-polygon with more than three vertices and the immense overhead associated with vertex reordering.

The second one is to chase clusters in adjacent 3D subspaces. When using the basic 3D subspace exploration mode we frequently observed that interesting patterns were starting to evolve but their full exposure was out of reach since it occurred in a different, albeit nearby, subspace. In these situations we often felt the need to "break out" of the current 3D subspace in the direction of the trackball movement such that these patterns could be reached. To solve this shortcoming we added a smooth subspace transition capability. It allows users to interactively change the influence of the data dimensions whose projections align with the current trackball movement and so increase their bias in the projection matrix P. This gives the exploring user access to the adjacent 3D subspace where the patterns of interest are better expressed. It lets him/her explore the data with a higher emphasis on one or more attributes of interest.

To engage into this mode of exploration users would let go of the left button and instead press the right button while moving the mouse in the direction of the desired dimension's projection, as indicated by the corresponding attribute's label on the trackball's periphery. The further the mouse is moved the more the projection plane is tilted into the dimension's axis vector. Conversely, moving backwards along that direction, towards the center of the trackball, decreases the influence of this dimension.

As Fig. 11 illustrates, ideally, we would accomplish this task by adding (or subtracting) increments $\Delta x=k_a \cdot \Delta d \cdot \sin(\theta)$ and $\Delta y= k_a \cdot \Delta d \cdot \cos(\theta)$ to the PPA-x and PPA-y vectors, respectively, where θ is the angle between the mouse movement vector and the trackball x-axis (the PPA-x vector), Δd is the distance the mouse moved in the direction of the projected dimension axis vector (positive when moving towards the periphery, negative otherwise), and k_a is a user-adjustable speed constant (we use dot products instead of the trigonometric functions). Subsequently, Gram-Schmidt is used to re-orthonormalize P (see Section 4.1), using the original PPA z-axis vector. One problem here is that, after Gram-Schmidt, the direction of this data dimension would change and there might be other dimensions taking the selected one's direction. We overcome this by fixing the selected dimension until the user releases the mouse.

This basic approach generalizes to more than one dimension. Fig. 4 illustrates this practical case in which there might be two or more projected dimension axis vectors in close range of the

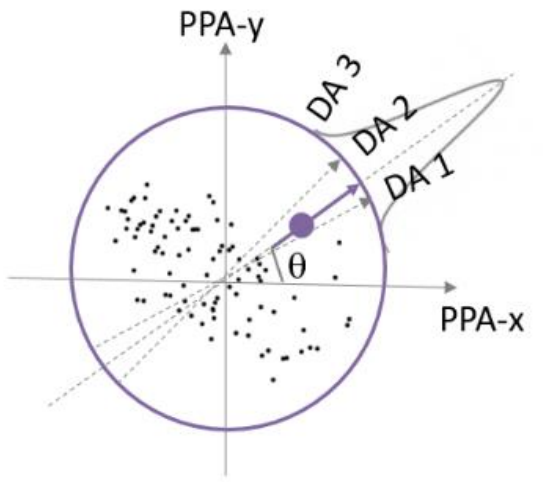


Fig. 11. Updating the PPA x-axis and PPA y-axis vectors by moving the mouse towards one or more dimensions. The influence of each dimension is weighted by a Gaussian function

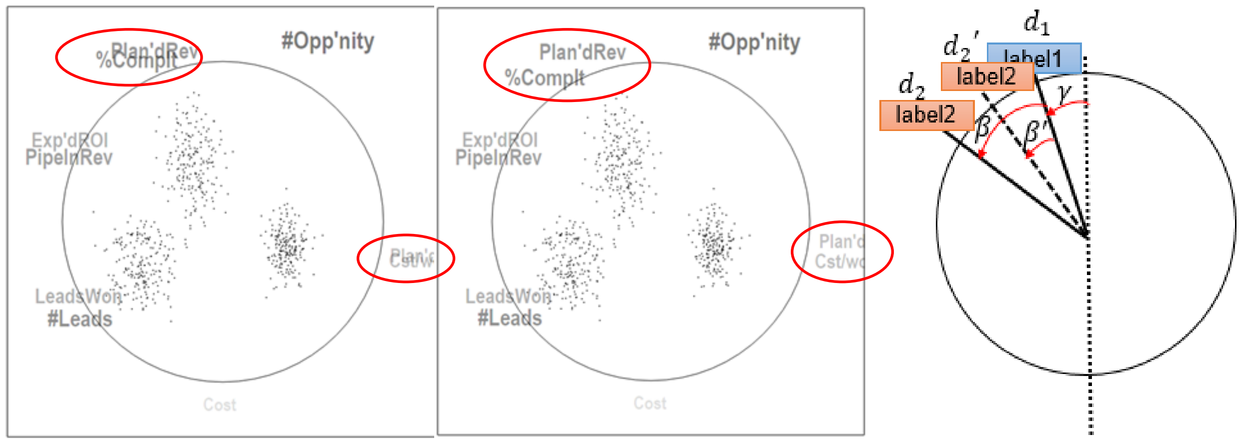
exploration direction. This might be an indication of multivariate relationships. To properly scale the axes vector influences geometrically, we apply a Gaussian weighting in terms of their direction misalignment. This is done via the following equation: $w_d=\exp(-k_d \cdot dot(v_m, v_d))$ where w_d is the weight applied to this axis vector, v_m and v_d are the direction vectors of the mouse and the axis vector, respectively, and k_d determines the reach of the Gaussian. The remaining steps are as for the single-vector case described in the previous paragraphs.

Our system also supports the case in which a user would first select an attribute via mouse click on the trackball boundary but then move the mouse in a direction not necessarily aligned with the attribute's dimension vector. This will gradually align the dimension vector with the mouse movement and move the attribute label accordingly. Again, the selected dimension's weighting changes according to the direction and length of the mouse movement.

The third one is to let users to go deeper into high-dimensional space. By clicking the middle mouse button, our system generates a PPA-z vector according to the three options described in Section 3.3.1. Then, based on the current PPA-x and PPA-y vectors – the first two rows in the compound matrix M – a new orthogonal vector is computed using Gram-Schmidt. This might at first glance not change the current projection at all, but when combined with trackball movements it directs users to a new 3D space, essentially going deeper into the high-dimensional universe.

4.3.4 Display of Attribute labels on the SE Boundary

In order to better comprehend the relationships between a scatterplot projection and the data dimensions (attributes), we display the attribute names as labels along the SE trackball periphery (see Fig. 12(b)). How much a dimension contributes to the projected point cloud is indicated by



(a) (b) (c)

Fig. 12. Dimension label overlap prevention. (a) With label overlap. (b) Without label overlap. (c) Illustration of this scheme

label size and opacity. The larger and bolder the label's font is, the stronger is the attribute's contribution in the plot. The location of each label is computed by the attribute's weighting in the PPA-x and PPA-y vectors. Let w_x be the PPA-x weighting, and w_y be the PPA-y weighting. Then the angle of this dimension vector and the positive x-axis direction is computed as $\alpha = \operatorname{atan}(w_y/w_x)$.

In practice, attribute labels may come to print on top of one another (Fig. 12(a)). This occurs because during trackball movement the weightings of dimensions constantly change, and very commonly several dimension vectors overlap. We have solved this problem by forcing labels to locate at least β degrees apart from their neighbors. Fig. 12(c) shows this for the upper left quadrant where d_1 is the location of label1 located γ degrees away from PPA-y and d_2' is the location of neighboring label2, spaced β' degrees away. We see that β' is too small causing the two labels to overlap. Therefore we introduce a small displacement which places label2 atd_2. Now label1 and label2 are spaced β degrees apart and no longer overlap.

In experiments we found that the best choice for β is dependent on the orientation of the dimension vector. The more vertical it is, the larger β should be, while for a more horizontal alignment, a smaller β will suffice. The logic behind this is simple – when labels are displayed on the top or bottom of the trackball, since the text is displayed horizontally, they need a larger distance for better separation. But when they are displayed on the left or right, only a small misalignment will be sufficient for readability. We therefore assign β a value dependent on the dimension vector orientation. Specifically, we choose the following piecewise linear function which relates β to the angle γ between the vertical axis PPA-y and the dimension vector (this equation is valid for the upper left quadrant only – the other three quadrants are related by symmetry):

$$\beta = \begin{cases} \theta_v - (\theta_v - \theta_h) * \dfrac{\gamma}{45°} & 0 \leq \gamma < 45° \\ \theta_h & 45° < \gamma \leq 90° \end{cases}$$

Here, θ_v and θ_h are constants we determined for the maximal font size of the labels which occur when the corresponding dimension vectors are fully projected. The angle θ_h = 4° is the displacement needed when $\gamma > 45°$, while an angle of $\theta_h = 24°$ is needed when γ=0°. When γ is between 0° and 45° we determine β via linear interpolation. Fig. 12(b) shows the configuration of Fig. 12(a) with our label displacement scheme enabled.

When using the system we made several additional observations. First, we found that while displacing the labels in the way described above provided for better readability, it was distracting in interactive mode when the user was rotating the trackball, since it could lead to sudden jumps of the labels. Hence we only apply the overlap removal method when the projection is fixed (when the user has released the mouse). Second, when a dataset has many dimensions, the label overlap can never be prevented. For this reason we added a slider to the control panel by which users can set the maximum number of attribute labels that are being displayed. Via the control panel, users can also choose to keep the labels of the 3D subspace's most significant attributes, or they can click on dimension names while pressing down <ctrl> to select the attributes they wish to track.

4.3.5 Point Brushing, Tagging and De-Activation

Our interface also provides the ability to label a point, or groups of points, with a color chosen from a palette. This is useful when monitoring a certain point or point group's behavior when the trackball rotates. It greatly helps in distinguishing different clusters or seeing sub-clusters emerge during motion.

Conversely, by painting a selected group of points in gray they will become inactive and are excluded from all further analysis, such as clustering and others. They can also be made invisible (by checking the 'Turn Off' checkbox in the control panel) which helps in recognizing other structures that were hidden or ambiguous before this removal. Finally, it is also helpful if the dataset contains outliers.

4.4 THE SUBSPACE TRAIL MAP

The subspace trail map (STM) serves three purposes: (1) it enables users to keep track of the subspaces explored so far, which they can revisit for further exploration, (2) it serves as a presentation platform for the system to suggest new subspaces not yet explored; and (3) it permits users to define routes along which they can transition between two or more of these subspaces, essentially using them as key frames. Users can double click anywhere on the STM to add the current view in the SE. For clustered data, if users want to add all subspaces where each cluster reside, they can simply click the ‘AllSubspace’ button.

4.4.1 Mapping Subspaces to the Subspace Trail Map

The STM is configured as an (invisible) irregular N-sided polygon with the N data dimensions mapped to its vertices, called the *STM-frame*. In this map, an added subspace appears as a small 2D scatterplot projection. We use PCA to place those small views in order to spread them out. In case of overlapping, the ‘SmallViewSize’ slider can be used to change the size of the small views.

Let’s assume there are p small scatterplots in the STM and the dimensionality of the data set is N. The compositions of the three PPA axes of the p subspaces can be formally represented as follows:

$$PPA_{ij} = \sum_{k=0}^{N-1} w_{ijk} * d_k$$

In this equation, i is either x , y or z and j varies from 0 to $(p-1)$. PPA_{ij} represents either PPA-x, PPA-y or PPA-z axis of the j^{th} subspaces. Similarly, w_{ijk} means the weighting of the k^{th} data dimension on PPA_{ij} and d_k is the k^{th} dimension. We then use the L-2 norm to define the overall weighting of the k^{th} data dimension for the j^{th} subspace:

$$W_{kj} = \sqrt{w_{xj} + w_{yj} + w_{zj}}$$

We then obtain a N dimensional vector for each subspace, representing the overall weightings of all data dimensions on it:

$$S_j = \left[W_{0j}, W_{1j}, W_{2j} \dots W_{p-1,j}\right]$$

We treat each subspace as an N dimensional points and perform PCA on all of them. We keep the first 2 PCs and project all points (subspaces) onto them. Since PCA automatically seeks to find the directions that maximize the variance of the data points, the small views representing the subspaces would be organized in a way that overlapping is reduced.

In case of inevitable overlapping due to the fact that some subspaces share very similar compositions, we provide the users with two methods to overcome the clutter. The first one is to put the mouse over the hidden view and it will be brought to the front. The second one is to use the “SmallViewSize” slider to change the sizes of the small scatterplots and by reducing the sizes, hidden views can also be revealed.

The STM is surrounded by a word cloud of dimension labels. To prevent a clutter of words we only show labels of the top L most significant dimensions where L is user-definable. The labels are placed based on the PCA results on the saved views.

4.4.2 Subspace and View Optimization

We perform view optimization on two levels: (1) to produce an optimized 3D subspace from a higher-dimensional one (as just mentioned for the subspace clustering), and (2) to automatically generate a salient 2D view for insertion into the VG.

Apart from the generation of novel subspaces, both optimization modules can also be used ad-hoc during exploration. For example, when chasing clusters into neighboring subspaces, users might employ view optimization within a narrow range of dimension updates to optimize the search. Another, more frequent application of this module aids users in the trackball-based exploration, accelerating the tedious manual exploration needed to find a view that fits a certain criterion, such as cluster separation.

A popular view optimization method in the context of high-dimensional data visualization is projection pursuit. Starting from any projection, projection pursuit returns the PPA x-axis and PPA y-axis that optimizes the targeted projection pursuit index (PPI). A number of methodologies have been proposed for this task – hill-climbing [16], random search [74], or modified simulated annealing [17]. We have strived for a sophisticated yet comparably easy-to-implement algorithm – Ant Colony Optimization (ACO) [19]. To the best of our knowledge ACO has not been used for the task of projection pursuit so far. And so we believe that its ease of implementation might be helpful to the readers of this article.

ACO simulates the behavior of ants in nature. When looking for food, ants initially travel randomly until they find food. On their way back they leave a pheromone trace along the route. Instinct prescribes that other ants most likely follow this pheromone trace instead of wandering randomly. But pheromone also evaporates gradually, and so, over time, shorter paths will be travelled more frequently, become more attractive, and therefore most ants will choose this path. Based on this intuition, the simplest ACO algorithm consists of the following three steps executed iteratively: (1) construct solutions, (2) evaluate solutions, and (3) update pheromone. It has been shown that the solution so generated is typically quite close to the optimal solution, if not identical to it.

The ACO algorithm requires a discrete search space and an objective function. Every time solutions are constructed, only those ants with good scores (higher/lower objective function values) can increase the pheromone amount along the path thus increasing the probability that the same path is chosen in the next iteration.

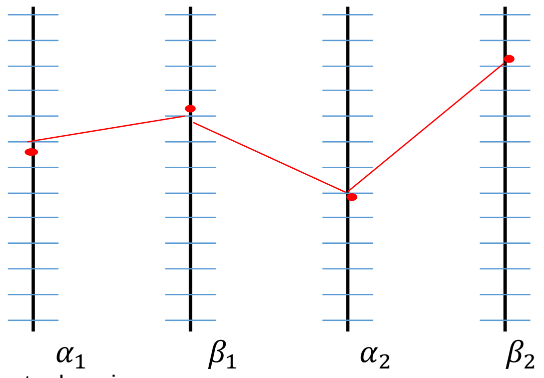
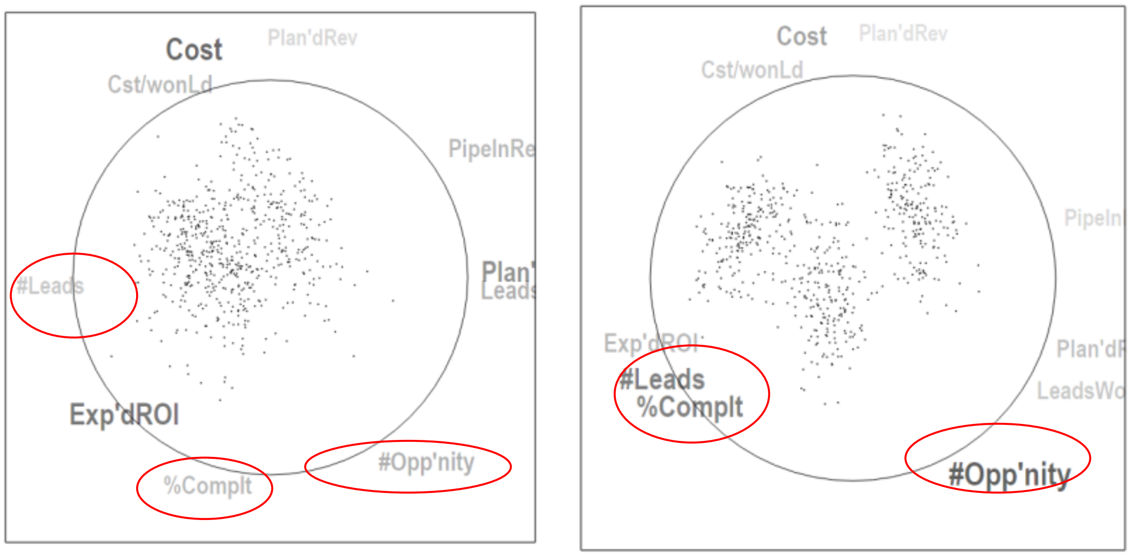


Fig. 13. The ACO algorithm in the discrete domain

The search space is the set of all possible PPA x-axes and PPA y-axes. The objective function in our case is straightforward – the chosen PPI. For example, for MDS the PPI is to minimize the normalized sum of differences between pairwise distances in the high-D and the low-D spaces – the stress. But really any quality criterion can be evaluated as long as it can be scored by a numerical value. This makes it quite easy to "plug-in" new quality metrics, even on the fly during exploration.

A small caveat is that projection pursuit is typically performed in the continuous domain, while ACO is inherently a discrete-space algorithm. Solutions to this problem were proposed in [14][83]. We decided to use a grid-based approach. To explain, suppose we only have a 2D data set and the PPA x-axis and PPA y-axis can be represented as: $PPA_x = \alpha_1 d_1 + \beta_1 d_2$, and $PPA_y = \alpha_2 d_1 + \beta_2 d_2$. There are four unknowns altogether - $\alpha_1, \beta_1, \alpha_2$ and β_2. We use four vertical gridded bars of Fig. 13 to denote those unknown parameters.

Our ACO algorithm differs from the traditional one in the selection of the starting path. While the traditional ACO typically begins with a random path, ours cannot because we begin from an initial PPA x-axis and PPA y-axis configuration, e.g., a randomized view. We start instead with the closest path– the red path in Fig. 9. In our example, suppose the starting view's PPA axes are: ${PPA_x}^{curr} = {\alpha_1}^{curr} d_1 + {\beta_1}^{curr} d_2$, and ${PPA_y}^{curr} = {\alpha_2}^{curr} d_1 + {\beta_2}^{curr} d_2$. Our ants would start from the red path which intersects the four vertical lines on ${\alpha_1}^{curr}, {\beta_1}^{curr}$, ${\alpha_2}^{curr}$ and ${\beta_2}^{curr}$. This path would have higher pheromone to begin with. We can constrain the searching range on each dimension to be close to the starting path to perform a local optimization by setting the two ends of the vertical bars to be close to the initial values. All ants construct different paths according to the pheromone the paths have. After finding a path each time, they lay different amount of pheromone according to the quality of the path (PPI). Our ants stop after a fixed number of iterations and the path with the highest pheromone amount would be the solution. It is in fact a piecewise linear path across this grid, which looks similar to a discrete parallel coordinate display. Performing the ACO algorithm on these discrete values is feasible. It



(a) (b)

Fig. 14. Equally expressing several dimensions. (a) The original projection. (b) The optimized projection where %Complete, #Opportunity, and #Leads are equally expressed.

will eventually result in one or more narrow clusters of polylines which are the set of optimal solutions.

The above is what we use when finding subspaces for clusters since it guarantees that the found view is close to the origin. We can also loosen the condition and do a global search. Then the resulting projection would be a global optimum according to different criteria. Further, we should also take into account that our ACO algorithm's return values are PPA vectors, which need to be of unit length and orthogonal. We therefore always normalize the returned PPA x-axis and then use Gram-Schmidt to find the corresponding PPA y-axis.

Our system also allows users to select several dimensions and produce a view in which those dimensions are equally expressed. This is achieved by letting users click on the respective labels along the trackball while pressing down the ctrl- and space-keys. Then, when releasing the mouse, the weightings for those selected dimensions are set to the maximum. A Gram-Schmidt step follows to orthogonalize the transformation matrix. See Fig. 14 for an example.

Fig. 15 shows results we obtained with our ant-colony based subspace and view optimization framework, again for the sales campaign dataset. We first applied simple k-means clustering using the Structure-Based Distance Metric of Lee at al. [56] and found three subspace clusters. A subsequent PCA analysis for each cluster established three subspaces. Clicking the 'AllSubspace' button adds all three subspaces to the STM. Fig. 15 shows the STM map. We obtained these projections by projecting all data points into the top two PCs for each subspace. We colored the three clusters blue, magenta, and green, and the outline colors of each small view

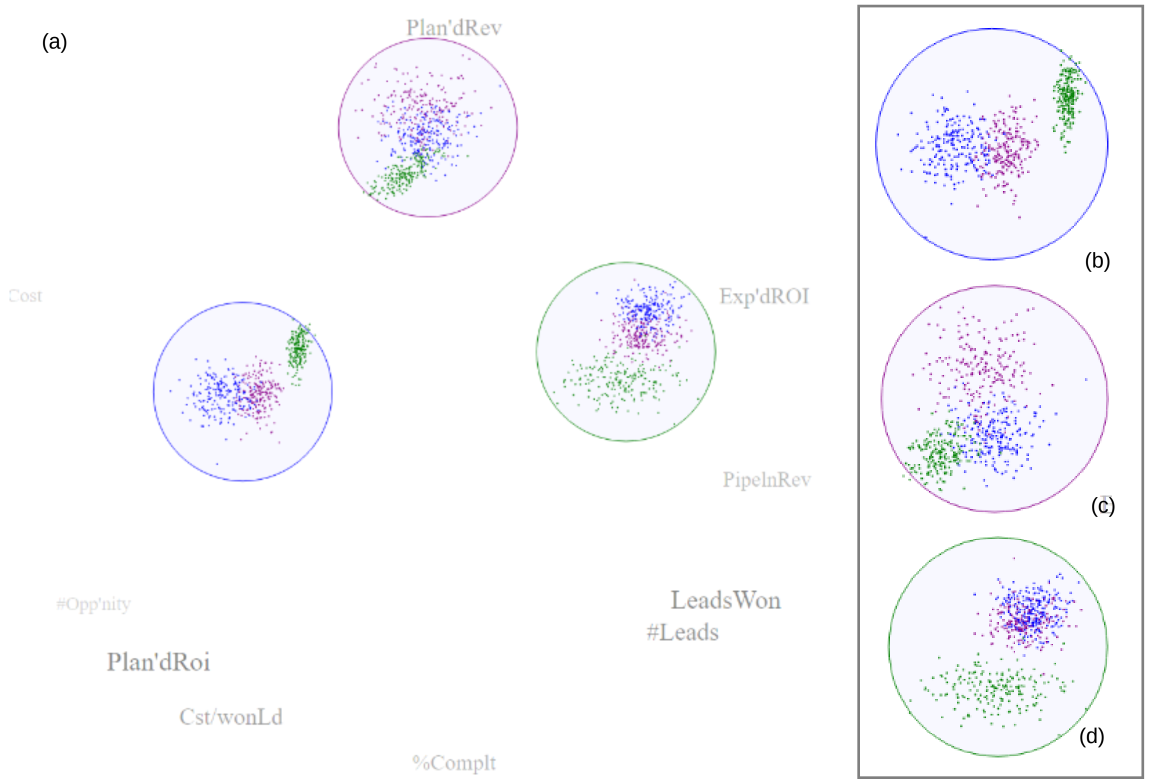


Fig. 15. PCA combined with the view optimizer to find separable subspaces. The respective subspace clusters are colored blue, magenta, and green. (a) The STM holds the three not optimized subspaces for the three clusters. We observe that the PCs alone cannot isolate the subspaces well – there is still a significant amount of cluster overlap. (b)(c)(d) Optimized subspaces for the blue, magenta and green cluster using the distribution consistent criteria. All subspace clusters are now well separated from the others in their respective subspaces

distinguish the corresponding cluster's subspace. Observe that by just projecting the points into this basis the clusters still overlap for all subspaces, but especially for the magenta subspace.

Next we optimized the subspaces found by PCA, using distribution consistent criteria [82]. The results are displayed in Fig. 15 (b)(c) and (d). We observe that the blue cluster's subspace is almost unchanged. This is because the three clusters are already well separated here. And since we only run optimization in a close range of the original PC projection this view might already be the best compared to its neighbors. (We might get better views if we optimize the view globally.) But conversely, the subspaces of the magenta and green clusters have significantly improved – in each panel the respective subspace clusters are now clearly separated from the others.

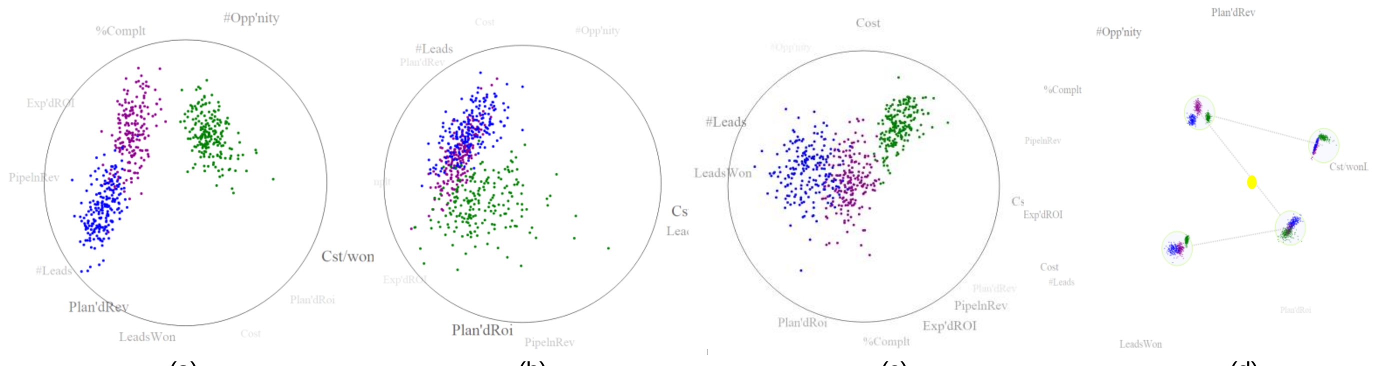


Fig. 16. Transitioning between two subspaces marked in the STM using the slider. (a)(b) and (c) are three intermediate views. (d)The defined path in the STM. The yellow dot is the indicator of where view in (b) is. Please see the provided video for the complete animation.

4.4.3 Transitioning Between Subspaces

Self-initiated and controlled animation can be a helpful paradigm for humans to understand how two or more different representations of the same information relate to one another [76]. We have employed animation to help users understand how two subspaces relate to one another, with the added aim that this might also instill a better understanding of the high-dimensional data space in a larger context. In our framework, users can select multiple subspaces in the STM, a path is then formed according to the order of small views being selected. Users can then manipulate the 'TraverseBtw' slider to change the PPA axis vectors from one subspace to another, and back.

However, simply linearly interpolating between PPA axes would lead to nonlinear intermediate projections. We adopted the algorithm by Cook et al. [17] to transition between the two subspaces using singular value decomposition.

Fig. 16 shows three snapshots of a sequence of frames from such an animation along with the path connecting the two corresponding nodes in the STM. In Fig. 16, panel (a) is the starting view, panel (b) is an intermediate view, and panel (c) shows the projection reached at the target view. Fig. 16 (d) shows the path in the STM. The yellow dot in the path indicates where the current view is. Since these still frames can only provide a limited illustration, the reader is encouraged to view the video to appreciate the insightful visual effect of this animation.

Alternatively, we also include a 'presentation mode' by letting the presenter to click the 'Next' button to go to the next key frame instead of using the slider. This provides a smooth transition between findings when representing the results to the audience, instead of abrupt changes of views.

4.5 CASE STUDIES

In this section we will demonstrate the versatility of our framework by ways of applying it in a diverse set of use scenarios involving high-dimensional data. The following sections will show our framework's application in (1) visual cluster analysis (2) visual item discovery and selection, helping users to recognize and negotiate tradeoffs among items, (3) visual cluster refinement, allowing users to partition feature-driven clusters based on the visual expression of the aggregation of these features， and (4) visual setup of a classifier in the presence of intermixing outliers.

4.5.1 VISUAL CLUSTER ANALYSIS

To illustrate the trackball interactions, we chose a multivariate cluster analysis task – an interactive study of the salesforce working for a large company. This discussion forms the first of four application scenarios we present in this paper. Our dataset consists of 900 points (one per salesperson) and 10 attributes parameterizing the basic corporate sales pipeline. Briefly, a sales campaign begins with a *leads* generator who produces prospective customers that a salesperson might be able to close a deal with. If these leads receive positive responses, they become *won leads* and receive a sales pitch at *cost per won lead*. Upon further positive response they become *opportunitie*s, or potential customers. *Cost* is involved in every step and high *pipeline revenue* is the ultimate goal.

Let's assume a sales team analyst, Pat, who is about to analyze the dataset. He begins to treat the whole dataset as one cluster and performed PCA on it. The PCA view of the whole dataset is shown in Fig. 17(a). He immediately notices that there are three visually separable clusters. Recall that there are three sales teams; this finding seems to be correct. To double check, Pat presses the 'Cluster' button in the control panel without entering a specific cluster number. This enables our system to run automatic K-means clustering powered by the elbow method to find the optimum cluster. The result (small view in the bottom right side of panel (a)) confirmed that those three clusters are real clusters. Pat now goes and examines the SE boundary in Fig. 17 (a), he notices that there are several group of attributes that have fairly strong correlations – *Expected ROI* and *Pipeline Revenue, LeadsWon* and *#Leads*. This gives him further insight into the dataset. Pat adds this view to the STM so he can easily return to this key finding.

Next, Pat wants to examine the subspaces of each cluster. He performs PCA on each cluster and adds all of them to the STM (Fig. 17(b)). Those three subspaces are optimized such that their corresponding clusters are better separated from the rest clusters. The blue, magenta and green outlined scatterplots are the subspaces for the blue, magenta and green cluster while the unlined one is the subspace for the whole data.

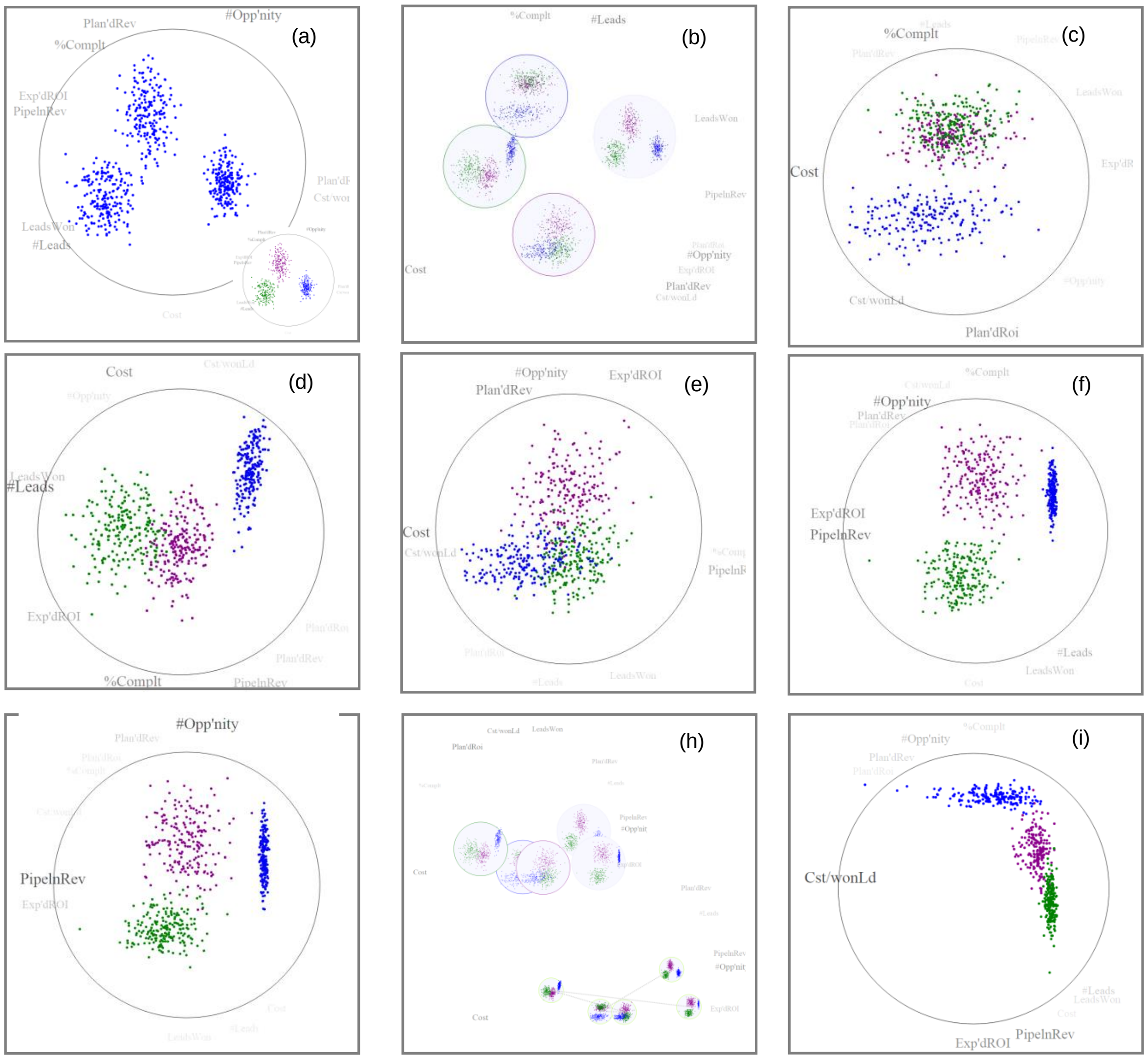


Fig. 17. Analyzing the sales force dataset. (a) The dataset projected on to the first two PCs. There are three visually separable clusters. K-means clustering algorithm using the elbow method confirmed that the three groups of data are real clusters. (b) Extracted subspaces for the three clusters. Those three subspaces are optimized such that their corresponding clusters are better separated from the rest clusters. They are added to the STM along with the initial PCA view. The blue, magenta and green outlined scatterplots are the subspaces for the blue, magenta and green cluster while the unlined one is the subspace for the whole data. (c) The subspace for the blue cluster. Cost is the attributes that has the highest variance for this group of data. *%Complete* and *PlannedROI* are the attributes that the blue cluster performed differently than the rest two clusters. (d) The subspace for the green cluster. *#Leads, Cost* and *%Complete* are the attributes that has the highest variance for this group of data. The green cluster generated a lot of *Leads* compared to the magenta and blue cluster. (e) The subspace for the magenta cluster. *PlannedRev, Cost* and *ExpectedROI* are the attributes that have the highest variance for this group of data. The combination of *PlannedRev, #Opportunity* and *ExpectedROI* separate the magenta cluster from the rest two. (f) Use the STM to bring the PCA view back to the SE and increase the weighting of *PipelineRevenue* and *ExpectedROI*. Both the green and magenta clusters generated more revenue than the blue cluster. The green cluster is slighter better than the magenta one. Note the *#Opportunity* is also fairly well expressed in this view and it shows that the purple cluster has higher number of opportunities than the green one. (g) Increase the weighting of #Opportunity along the PPA-y axis and produces a traditional bivariate scatterplot projection. (h)Top: Save this view to the STM and there are some views overlapping. Bottom: Reduce the size of the small views and reveal all views. (i) Increase the weighting of *Cost/WonLead*. The blue cluster assigned the highest Cost to each WonLead. Consider the fact that is also produced the least *PipelineRevenue*, it might not be a good idea for this strategy.

He first brings the blue cluster's subspace back to the SE for examination (Fig. 17(c)). He notices that *Cost,* which is at the left hand side of the PPA-x axis, has the biggest and boldest font and it indicates that this has the green clusters varies a lot on it. What's more, in the PPA-y direction, *%Complete* and *PlannedROI* are the attributes that the blue cluster performed differently than the rest two clusters – it has less complication rate but higher *PlannedROI*. Pat goes on and brings the subspace for the green cluster back to the SE (Fig. 17(d)). He notices that *#Leads, Cost* and *%Complete* are the attributes that has the highest variance for this group of data. Based on the size and location of the attribute *#Leads*, he can tell that the green cluster generates a lot of *Leads* compared to the magenta and blue cluster. Pat keeps his exploration by taking the subspace of the magenta cluster back to the SE (Fig. 17(e)). Similarly, he observes that *PlannedRev, Cost* and *ExpectedROI* are the attributes that have the highest variance for this group of data. The combination of *PlannedRev, #Opportunity* and *ExpectedROI* separate the magenta cluster from the rest two.

Pat knows that high *Pipeline Revenue* and *Expected ROI* are important targets for any business. He decides that it would be good now to explore how the company's salesforce compares with respect to these two revenue parameters.

He uses the STM to bring the initial PCA view (small panel in Fig. 17(a)) back to the SE. He presses the right mouse button and moves the mouse in the direction of the two attributes in Fig. 17(a) (upper left corner). Fig. 17(f) shows the outcome. We notice that the font of the two revenue labels gets stronger which means that the corresponding two attributes receive more weight in the viewed 3D subspace. This outcome shows that both the green and magenta clusters generate more revenue than the blue cluster. The green cluster is slighter better than the magenta one. Pat also notices that *#Opportunity* is fairly well expressed in this view and he decides to now cast this relationship into a traditional bivariate scatterplot by increasing the weighting of *pipeline revenue* along the x-direction, making it the PPA-x, while *#opportunity* is increased along PPA-y. He achieves this by selecting the label *pipeline revenue* with a mouse click and then moving the mouse with the right button depressed into the direction of the x-axis. Fig. 17(g) displays the outcome. It shows that while the green and magenta clusters vary in the number of opportunities – magenta creates more – both groups have somewhat similar revenue. On the other hand, the blue cluster has high *#Opportunity* but its revenue is low. There must be something else going on.

Pat saves this view in the STM (top panel in Fig. 17(h)) and continues to explore. Pat notices that the STM is much cluttered right now and he uses the "SmallViewSize" slider to reduce the size of the small views (bottom panel in Fig. 17(h)). Pat continues to explore the data. He always thinks that how much cost a sales team puts on a won lead should be one of the important sales strategies. This time, he makes *Cost per won lead* the PPA-x axis (Fig. 17(i)). He notices that the blue cluster assigned the highest Cost to each won lead.

Based on all the discoveries Pat makes, especially those in Fig. 17(d), (g) and (i). Pat concludes that the blue team doesn't have lots of Leads, but put a high cost on those won leads and they get lots of opportunities. Lots of opportunities sounds like a winning strategy but because of the high cost per won lead, their generated revenue is very low. On the contrary, the green team has lots of Leads and assigned low cost on those won leads. Even though they do not have lots of opportunities, their final revenue is still high.

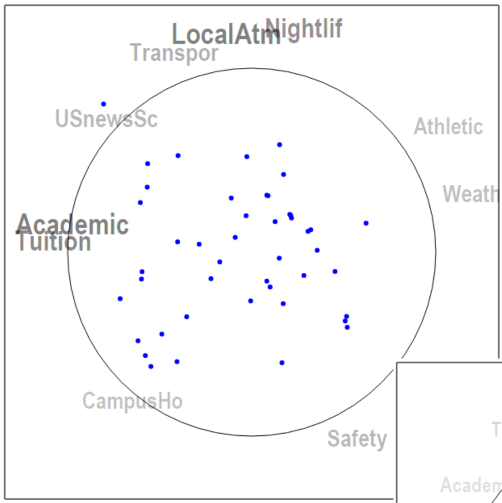
Pat also notices that even the number of leads the magenta team has is not as high as the green team and they both has low cost per won lead, the purple team produces more opportunities. He can continue to use our Subspace Voyager to explore the data and answer those questions

Pat is now very excited about his findings and he presents them to his co-worker Kate. Since he already saved all the key frames in the STM, he only needs to connect them by simple mouse clicking and build a path (bottom panel in Fig. 17(h)). Now, he only needs to click on the 'Next' button and all his findings would be displayed one after one, in a continuous animation fashion.

We believe that this example convincingly demonstrates how our SE interface enables users to playfully arrive at different multivariate scatterplot projections, quickly respond to new explorations ideas on a whim, make casual observations in the process, and just as easily return back to a traditional bivariate scatterplot visualization. The interested reader may watch the video to see the complete process. These proposed interactions can also be achieved by Radviz [36] when it allows users to change the weighting and positions of different data dimensions. This might be good to exam single projection but to our knowledge, there is no system that is based on Radvis and also use maps to save the findings. Our STM on the other hand allows users to record all important findings which they can use to present to audience. Radviz is also a non-linear projection which often leads to distortions of the data.

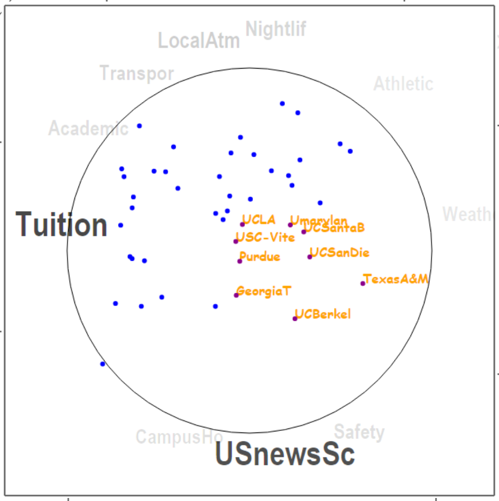
4.5.2 Visual Item Discovery & Selection

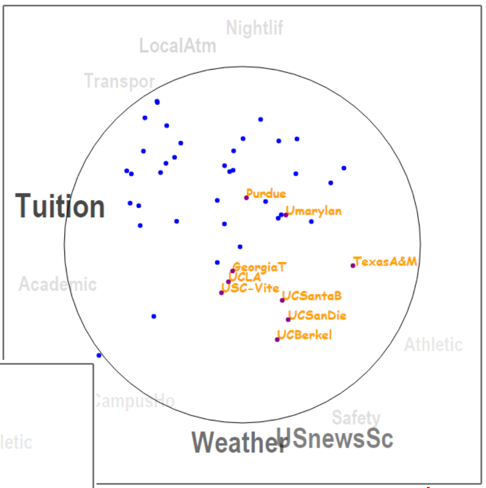
Selecting the best college, given the many personal constraints and preferences one might have, is arguably one of the most difficult choices a person will make in life. It involves the task of discovering the set of schools that best meet one's personal requirements, comparing them by weighing their trade-offs, and then selecting the college that fits best. Here we use the mixed dataset initially created by Nam and Mueller [69]. It has multi-faceted data on 50 of the top US colleges, enabling the college-seeking student to look at schools not only through the lens of academics, but also through the lens of social life and the general environment the school resides in. Academic ranking and tuition information were extracted from a leading source of such information – the US News & World Report [108]. The College Prowler website [107], on the other hand, ranks colleges on a multitude of social and environmental factors. We picked 8 of the 20 the site offers: athletics, campus housing, local atmosphere, nightlife, safety, transportation,



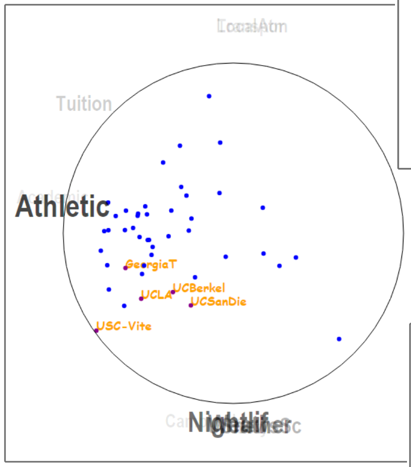
(a). The PCA view of the whole dataset. It reveals a strong positive correlation between Academic and Tuition, as well as between LocalAtmosphere, NightLifet and Transportation.

(b). Make Tuition the PPA-x axis and USnewsScore the PPA-y axis. The magenta-colored, labeled points are the schools with high US News Score and low Tuition.



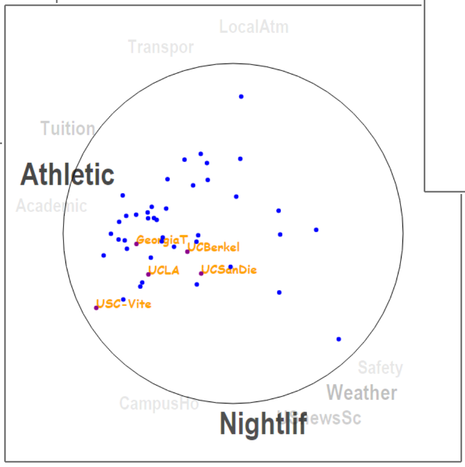


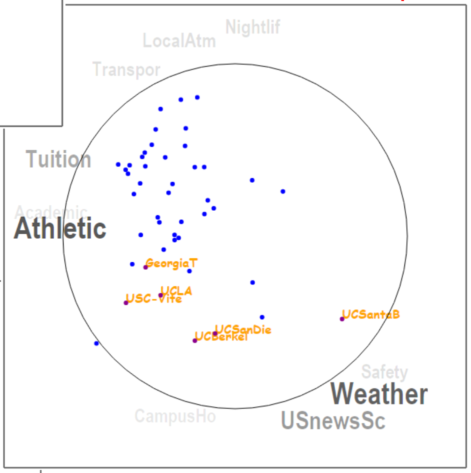
(c). Increase Weather along the PPA-y axis. Purdue moves up revealing its bad weather, TexaxA&M and UMaryland do not move much but are low in both scores.



(f). The final setup. The dimension projection reveals four dimension groups.

(e). Increase Nightlife along the PPA-y axis. USCViterbi has the best overall score, followed by UCLA, Georgia Tech, UCSanDiego and UC Berkeley.





(d). Increase *Athletic* along the PPA-x axis. UCSantaBabara has very low athletic score.

Fig. 18. Finding the best college in the college dataset.

academic environment, and weather. Each score is available letter-graded ranging from A+ to D-. We mapped these equidistantly to values in the range 0 to 1.

The College Prowler website allows users to navigate the space of college attributes by filtering – using slider bars and menu selections for each parameter to narrow down the search. This can be rather tedious and it also makes it difficult to recognize tradeoffs. We believe that our SE provides a more playful and targeted experience, with the STM serving a platform to save any intermediate findings.

In the following we shall follow 17-year old Tina, who is just about to finish high school, and see how she uses our subspace voyager to find the university she feels best about.

Tina starts out with a view onto the dataset as a single cluster using the primary PC axes as a basis (Fig. 18(a)). As mentioned in Section 3.2, in such a view the dimension vectors of strongly positively correlated attributes tend to coincide and as a result their labels map to similar locations along the trackball boundary. Conversely, negatively correlated attributes will map to opposite sides of the trackball boundary. The only condition for both is that their projection into the PC-axes basis is sufficiently significant, which is visually expressed in our system by a large and heavy label font.

In the initial view of Fig. 18(a) Tina observes two sets of positively correlated attributes: (1) *Academics* and *Tuition*, and (2) *LocalAtmosphere*, *NightLife* and *Transportation*. She also observes a few negatively correlated attributes, among them: (1) *Academic* with *Weather* and *Athletics,* and (2) *LocalAtomosphere* and *NightLife* with *Safety*. From these constellations Tina quickly recognizes that top academic universities tend to charge higher tuition, but at the same time their athletic teams are not necessarily among the best. She also learns that universities built in nice town or city areas usually have better night life and transportation systems, but they also tend to be less safe. All this is good to know before engaging into the actual selection process described next.

Tina does not come from a wealthy family and so her immediate focus is tuition cost. Her first step is therefore to select *Tuition* and move the mouse towards that label (to the left). Next she wants to see which of the schools have good academic ranking. She selects *USNewsScore* and moves the mouse downwards to maximize the spread. This leaves her with the axis-aligned scatterplot shown in Fig. 18(b). In this plot, all points on the lower right side are the universities with high rankings but low tuition – these are the ones Tina is interested in the most. She colors them in magenta and asks the system to label them (with the university names).

Tina likes the outdoors a lot which requires the weather to be generally good. So she adds *Weather* as another requirement to *USNewsScore* by selecting the *Weather* label and moving the mouse in the downward direction (Fig. 18(c)). This enables her to appreciate any tradeoffs that may exist in these two variables. After making this choice, she sees 'Purdue' moving away significantly. This means that even though its *USNewsScore* is quite good, the score is not good enough to make up for the unfavorable weather. Likewise, 'GeorgiaTech' also moves away but not as much and so Tina keeps it marked and labeled. 'UMaryland' and 'TexasA&M' did not move too much either, but both of their scores are not high to begin with. It means that none of these two schools does well enough in either *UsnewScore* or *Weather* to make up for the moderate performance in the other attribute. Tina removes these two schools as well, repainting them to neutral blue.

Tina enjoys the excitement of college team sports. She is also quite athletic and she thinks she might be able to secure an athletic scholarship to pay for the tuition. So she puts the *Athletics* attribute near the tuition using the aforementioned mouse interactions (Fig. 18(d)). She notices that 'UCSantaBarbara' has a rather poor athletic score and henceforth she eliminates that school. On the other hand, 'USC-Viterbi' has the highest athletic score, followed by 'GeorgiaTech', 'UCLA', 'UCBerkely' and 'UCSanDiego'. She keeps all them magenta colored and labeled.Of course, Tina wants to have some fun in college. She focuses on *NightLife* and moves it to the bottom (Fig. 18(e)). 'USCViterbi' moves down confirming that it has good nightlife. 'UCBerkeley' and 'UCSanDiego' move up, indicating that they may have good weather but the nightlife is limited. On the other hand, 'GeorgiaTech' and 'UCLA' stay put – they are more balanced in those two factors.

Looking at the plot shown in Fig. 18(e) Tina sees that 'USC-Viterbi' might be the best candidate, but it also has the highest tuition (Fig. 18(a)). On the other hand, 'GeorgiaTech', 'UCLA', 'UCBerkeley' and 'UCSanDiego' all could be possible candidates. In order to gain an overall impression Tina puts all attributes of interest into one view. She tilts the trackball and creates the configuration shown in Fig. 18(f). It shows four dimension groups: (1) *Athletic* and faintly *Academic,* (2) *Tuition,* (3) *LocalAtmosphere* and *Transportation* (both with small weighting), (4) *NightLife*, *Weather* and faintly *USNewsScore*, *Safety* and *CampusHousing*. Among all those groups of factors, Tina values *Athletic* and *Academic* the most, and so she chooses 'GeorgiaTech' as her #1 top choice school to apply for.

We purposely conducted a similar selection task than Nam and Mueller in [69], and a partial goal of this use case was to compare these two systems. We obtained rather similar, almost identical result than these authors, except that their final candidate list did not contain 'UCLA'. We compared UCLA's scores with that of the other candidates (see Table 1) and found that except for a lower rating in transportation and a slightly lower rating in *USNewsScore,* it is not worse in other aspects and hence should be included in the final candidate set. Especially in the final dimension group *Academic* and *Athletic,* it has a better combination than the other schools in the set, except for 'Georgia Tech'.

Table 1. Rankings of the final five candidates

College	Acad.	Athletics	Hous.	Atmos.	Night Life	Safety	Trans.	Weather	US News	Tuition
UCLA	10	10	5	12	10	9	4	11	69	22428
USC-Viterbi	10	2	8	11	12	7	8	11	77	22734
Georgia Tech	10	11	5	10	9	7	7	8	86	22188
UC Berkeley	10	9	8	9	8	7	10	11	89	14998
UC San Diego	9	8	6	11	9	11	6	12	72	14694

We think that the omission of ‘UCLA’ occurs because TripAdvisorND’s motion trail makes it sometimes difficult to precisely follow each point. But the motion trail is needed there to convey the dynamic movement. Conversely, with our trackball, motion trails are not required since the perception of the motion is much more tightly linked to the interaction that is causing it – both occur in the same interface. Another advantage of our new system is that users no longer need to take their eyes off the visualization while they are interacting to change the view while TripAdvisor’s touchpad required this. It also required that two points are moved separately, the one due to the PPA-x and the one due to the PPA-y vector. With the trackball interface presented here users can express these goals much more directly. In fact, they do not even need to be aware of the existence of these axes and vectors which we believe makes our interface much more appealing to general users.

Admittedly, similar reasoning can also be achieved by using LineUp [30]. However, LineUp needs users to manually input specific weightings for each attributes and compute the score as a weighted sum. It's not easy to achieve a weighing combination that fits the needs of the users. For example, Tina here would need to know “I want to give tuition 0.3, US News Score 0.4, weather 0.2, sports 0.1”. But does this combination guarantees that the final ranking is what Tina wants? To use our system, users do not need to know this specific composition and can explore the data in a playful fashion. LineUp also displays the data in a table style and users may need to scroll up and down to monitor the effects of different weighting combinations. Our system on the other hand displays all the data in one area (the trackball) and users can monitor them all at the same time.

4.5.3 Visual Cluster Refinement

Often high-dimensional data are derived from feature analysis where the features themselves are not overly meaningful in isolation. Rather, it is the synthesis of all features that allow users to describe a grouping of the data points, with the feature-based clustering providing the organization in which the boundaries of the individual groups can be delineated. In this final use

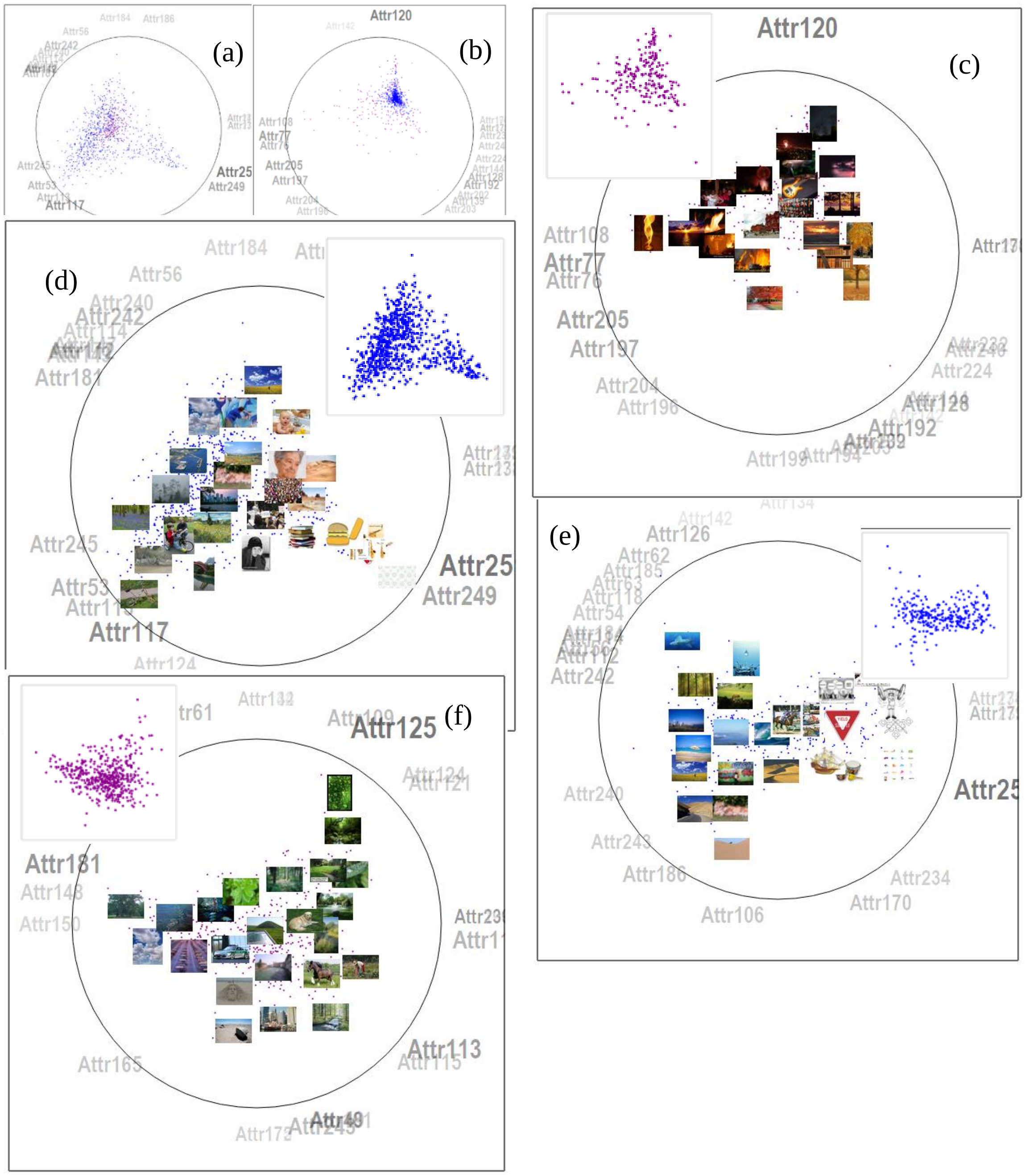


Fig. 19. Exploring the ImageCLEF dataset. (a) The subspace of the blue cluster. (b) The subspace of the dark magenta cluster. (c) The images in the dark magenta cluster. (d) The images in the blue cluster. (e) The images in the first sub-cluster in the blue cluster. (2) The images in the second sub-cluster in the blue cluster.

case we demonstrate how our system can be used to allow humans to assist in deriving and refining these kinds of groupings in data, using visualization as a gateway. We call this process *visual cluster refinement*.

We have selected an image classification tasks for this use case. The CLEF Cross Language Image Retrieval Track (ImageCLEF) [109], launched in 2003, is an evaluation forum for the cross-language annotation and retrieval of images. It aims at providing a cross-language and language independent platform where visual information retrieval systems operate, in order to assist apparent needs in visual media analysis, classification and retrieval. The ImageCLEF data [28][110] provide users three sets of images – training, testing and development. Each set uses different feature descriptors to describe the images, such as SIFT, colorhistogram, and GIST. We use the GETLF feature vector from the development set of ImageCLEF 2013 [111], which is a 256-dimensional histogram based feature. For the exact information on how to generate these descriptors the reader is referred to [28].

We now employ our subspace voyager as a medium to bring users into the loop of assessing and assisting the process of top-down clustering of this dataset. Since the cognitive processes driving image recognition and assessment are still much better developed in humans, as opposed to machines, a visual interface that allows humans to participate in this task can be very valuable. We begin by setting the initial number of clusters to a value of 2 and run k-means clustering with the structural based distance metric [56] on the collection of points. Fig. 19(a)(b) shows the two subspaces in which the two clusters reside. Since the attribute labels on the trackball boundary are rather cryptic, a visual quality is difficult and even more so is their interactive refinement. This can only be done by visualizing the semantics of the data points themselves – in this case the underlying images.

To facilitate this, we examine the two clusters separately inside their own subspaces by turning off the other cluster. We then randomly select a subset of the data points in each cluster and display the corresponding images next to them. Fig. 19(c)(d) are the displays for the blue and magenta clusters, respectively, and the small overlays on the side are the projections without the images. We notice that the images in the magenta cluster (Fig. 19(c)) are overall more saturated than those in the blue cluster (Fig. 19(d)). Their hue values are also different. Fig. 19(c) has more yellow (lots of fire and sunset, trees with yellow leaves) and black while Fig. 19(d) has more blue, green and gray. In the direction of the PPA-y, the images in Fig. 12 (c) gradually change from yellow to black, with some slight red tone in the middle. Even within the bottom yellow-toned images, the yellow in the left images is more intense than the yellow in the right images. In Fig. 19(d), the bottom right images are paler green mixed with gray while those on the top are mostly blue toned. The images on the right half have mostly white background, with the main objects being low saturated. The differences between the two clusters are obvious and this confirms that k-means separated the dataset well.

With these cluster visualizations in place we now continue this process by further partitioning each cluster. We start with the blue cluster by painting all magenta points in gray. This allows k-means to only run on the blue cluster. Fig. 19(e)(f) are the results. We again display a random collection of images in each sub-cluster's subspaces and the point-only projections along the side. We notice that the blue cluster (Fig. 19(e)) is blue tone based while the magenta cluster (Fig. 19(f)) is mostly green. The blue cluster also has some white images in its right half which points to the opportunity for further partitioning this cluster. This process might be continued until no further significant variations inside each clusters can be found.

4.5.4 Visual Setup of a Classifier

In this example we show how our system deals with very high-dimensional datasets, assisted by its integrated view optimizer suite. We use the data published for the CADASTER challenge [112]. It applies the MOE descriptor [113] to describe the structure of molecules. The challenge is to use the descriptor to predict the molecule's environmental toxicity. The toxicity information is given and so the data can be used to train such a predictor. This dataset contains 644 points, each 254-dimensional.

Since the attribute of toxicity is continuous, the solution would involve some form of regression. We changed the challenge into a binary prediction task, that is, determine of toxicity is positive or negative. This is similar to the approach of Yuan et al. [101] but they used a different descriptor of 221 dimensions and they only classified a portion of the data with approximate results. Yuan et al. tackled the problem largely via manual interaction in both the data and the dimension domains without much assistance by optimization schemes. Their classifier is based on a set of data dimensions, while ours is based on a set of hyperplanes which is a more general spatial descriptor. Our hyperplanes separate the data according to toxicity being positive or negative. Then, using those planes, new molecules could be classified as well.

Since we already know the class labels, we can begin with the subspaces of the two classes (Fig. 20(a)(b)). Unfortunately, even after view optimization, the two clusters still overlap in both views. In normal operation when using the view optimizer to find subspaces, we restrain the range to perform a 'local' optimization which ensures the optimized views are very close to the original PCA views. In this use case, we only care about finding views where the two classes are mostly separated, but those views do not necessarily have to be close to the current view. Thus, we relax the constraint and run the view optimizer 'globally' by setting the search range for all dimensions as [-1, 1]. Then, each time we run the view optimizer it starts with the current view and returns a view which separates the two classes better than the current one. Win this way we run the view optimizer a couple times until it no longer returns any better view.

Fig. 20(c) is the result after running the view optimizer on the subspace of the magenta class. We notice except for the middle area (in the red circle), the blue and the magenta classes are well separated by the green hyperplane. We record this information and then paint the well separated points in grey to further perform view optimization only on the circled points (Fig. 20(d)).

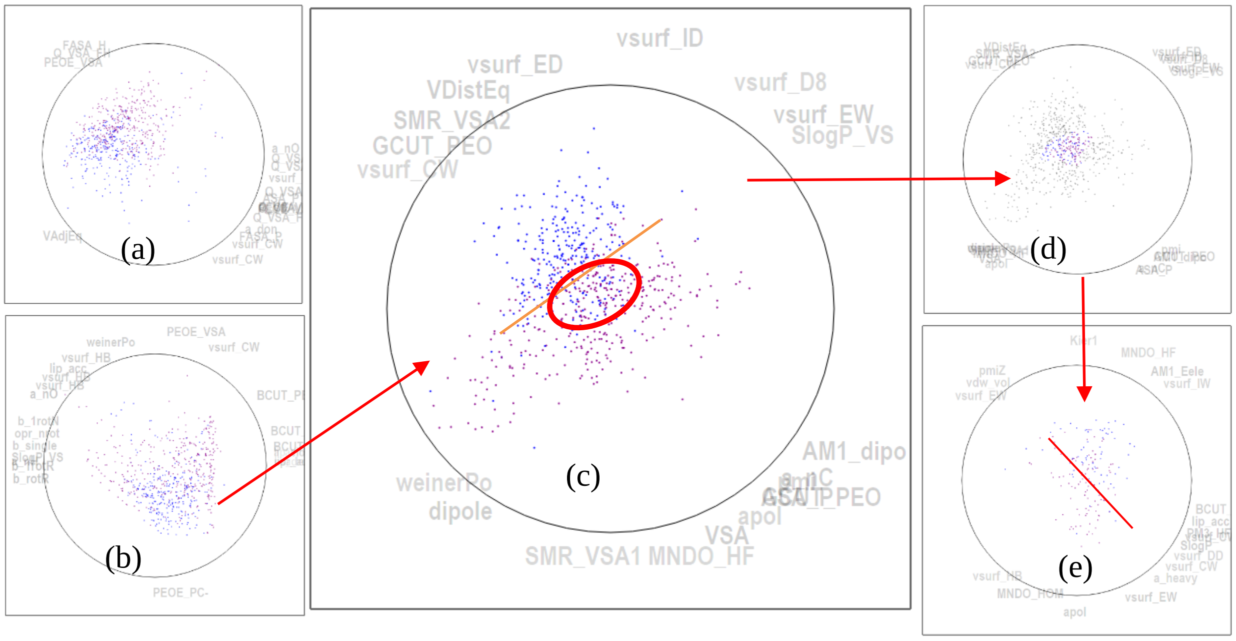


Fig. 20. Visual classifier setup using the MOE dataset (a). The subspace of the blue class (b) The subspace of the magenta class.(c).Run the view optimizer globally on the subspace of the magenta class -- except for the points inside the red circle, all others are well separated by the green hyperplane. (d) Paint the well separated points gray and run the view optimizer again on the rest of the points. This view only shows the active (non-gray) points. The orange hyperplane separates the two classes

.

Fig. 20(e) is what we get after a couple ACO runs (The gray points have been omitted for clarity). We observe that the two classes are well separated using the orange hyperplane except for a couple magenta points falling into the blue class region. We could focus on these points and run the view optimizer on them, but the amount of misclassified points is already rather small. But this is the advantage of our visual interface – users can decide where they would like to refine the classifier.

This use case shows that our approach of fusing visual interaction with view optimization can produce fairly accurate classifiers with just a few interactions. A new point would be classified by first determining if it projects into the red circle or not. If it does then the orange hyperplane would be used to classify it, else the green hyperplane.

4.6 USER STUDY

In order to evaluation the various interactions our Subspace Voyager provides, we conducted a user study among 10 graduate students. None of them had experience in visual analytics before. We wanted to test if the participants could fulfill certain data analysis tasks using the proposed interactions.

4.6.1 Setup

We invited all participants to sit down with us individually. At the beginning, we showed them a three minutes ten seconds long introductorily video which covered all the basic interactions the Subspace Voyager supports, such as the different trackball interactions, how to save views to the STM, how to bring the saved views back and how to traverse between the views. In that video, we used the Iris dataset [114] as a walking example.

We then let them ask us any questions about the system and this was the only time during the user study that we interfered. Three participants were a bit confused about how to interpret the dimensions' labels along the trackball at first but after a quick explanation, they all understood.

We then asked them to perform three tasks. We filmed the computer screen to record their operations and also asked them to speak out their thoughts during the whole time, which we also recorded. Both means of recording helped us analyzing the participants' performances later.

4.6.2 Tasks

There are three tasks we asked the participants to fulfill. For the first task, we used a 3D contrived data set which has a hollow tube with a stick in the middle, displayed both in the SPLOM (Fig. 21(a)) and our system (Fig. 21(b)). We at first asked the participants to describe the shape of the data based on the SPLOM and then asked them to use the Subspace Voyager to explore the data and describe it again.

For the second task, we initially put two snapshots (Fig. 22(a)) of the salesforce data set in the STM, asked the participants what they saw in the two scatterplots. We then asked them to traverse between the two and describe to us what they saw along the path.

The last task is a complete data analyzing one. We used the salesforce data set and showed the participants a 48 seconds long video to introduce this data. The initial trackball configuration is shown in Fig. 23 (a) where all three teams overlap. We then told the participants that there are three sales teams in this data set and they are colored differently in our system. Different teams use different sales strategies such as generating lots/a small amount of leads, assigning high/low cost per won lead and such. If the participant was the company leader, what strategies he/she would take that is more related to higher revenue generation. We asked the participants to explore the data first and after they done exploring, tell us their choices all together.

4.6.3 Results

In task one, because the structure of the data can only be observed from a non axis-aligned angle, not a single participant found the hidden stick from the SPLOM. 8 of them asked for pen and paper to help themselves construct the distribution of the data. They described the data as 'tilted

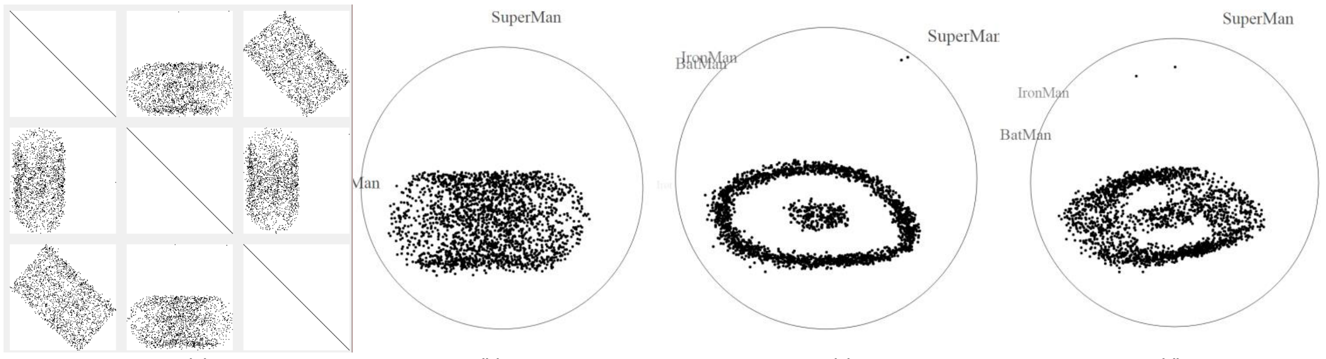


Fig. 21. User study 1. We asked the participants to user both SPLOM and our system to examine the shape of the data.(a) The SPLOM of the data. (b) The initial view in the SE. (c)(d) Typical views the participants generated that helped them draw the conclusion.

cylinder' or 'oval prism'. On the other hand, using our trackball's normal rotating functionality, all of them managed to find this hidden stick. They described the data as 'a pipe with something in the middle', 'cylinder with some coaxial cable', or 'two concentric cylinders'. Fig. 21(c) and (d) listed two typical views the participants generated that helped them draw the conclusion. This demonstrates that our trackball is able to help users understand the structure of the data.

We have a few interesting findings here. First of all, for the slowest participants, he initially only tilted the trackball in the upward or downward directions in Fig. 21(a) where the hidden structure cannot be revealed. After some time, he decided to tilt the trackball in the other directions and finally found the stick. He later told us that because of the SPLOM display, he did not suspect anything in the middle and therefore didn't intent to peek from that decisive angel. This makes us think in order to explore an unknown data set, sometimes it might help to not give too much prior information such that users would have the flexibility to perform all kinds of actions. Secondly, after finding the hidden structure, 4 users used the 'chase cluster' mode to re-produce the SPLOM in order to verify if the SPLOM is correct. This makes us think that our system might be a good tool for the users to verify known facts about the data.

In task two, 8 participants said that there are 2 clusters in this data set based on the two small snapshots. 2 participants suspected there being a third cluster based on the top view in Fig. 22(a). All of them used our 'TraverseBtw' slider to go from one view to the other (nine of them went from the top view to the bottom one while one of them went the other way). Top panels in Fig. 22(b)-(d) listed a few critical views along the path. It's hard to spot the three clusters without motion parallax in those frames, so for demonstration purpose, we colored the three clusters and put the same views below their uncolored counterparts here. All of the participants spotted the third cluster while travelling (for the two who suspected its existence in the beginning, they said they were not sure at first, but now they are sure). They described what they saw along the path as 'The bigger cluster separates into two, one of them remains a separate

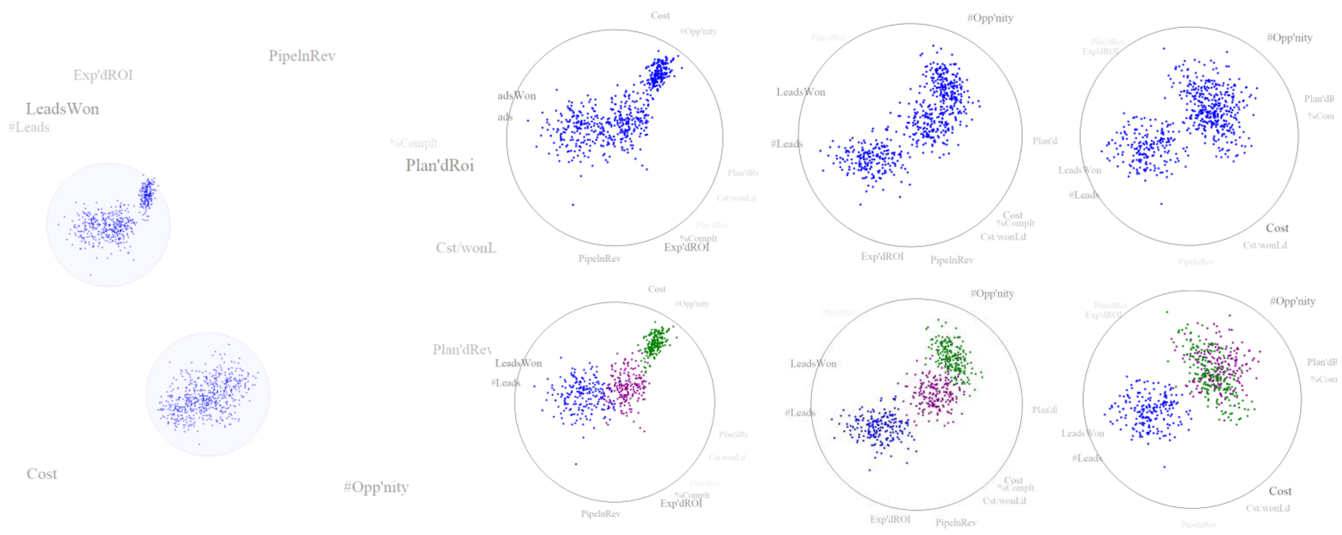


Fig. 22. User study 2. We asked the participants to use the STM to traverse from one view to the other and describe what they saw long the path.(a) The initial STM. (b)(c)(d).

cluster while the other one merges with the smaller cluster'. Some of them even described in more detail, 'The upper left cluster seems to be moving forward. Another cluster is moving upward and the third one is moving downward'. One guy also saved a couple views in the STM and later mentioned that, "if I look at those still frames, I probably still cannot tell that there are three clusters, it's really the motion that you can tell."

This task proves the use of our STM, especially the feature that allows users to traverse between frames. What's more, the power of motion parallax is also proved – users can spot patterns easier than relying purely on still frames.

For the third task The participants all used the 'chase cluster' functionality (directly move toward several dimensions' labels or first click on a dimension label and then move toward it to increase its weighting on the current projection) to analyze the data. But when it comes to the way they operated, three different strategies formed.

The first one is to make individual data dimensions be the dominant factor on either PPA-x or PPA-y axis, and observe the distribution of the three sales teams along these attributes. A typical view of this strategy is shown in Fig. 23(b). Here, the participant wanted to exam the influence of *#opportunity* and later on, when using the STM to go back to this view to present his findings, he described this way as 'The blue cluster has the lowest number of opportunities'.

The second one is to keep *PipelineRevenue* as either PPA-x or PPA-y axis, and make the rest attributes the other one. By doing this, those participants managed to create traditional axis-aligned scatterplots where they can exam the relationships between two variables. One typical

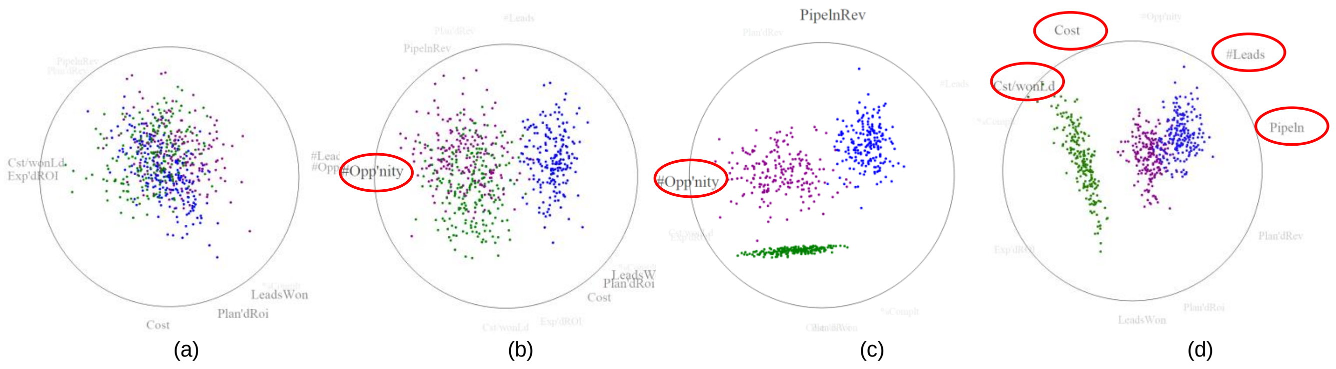


Fig. 23. User study 3. We asked the participants to analyze the sales force dataset.

view of this method is shown in Fig. 23(c). The participant can then conclude that “Blue and magenta clusters all have high revenue. Magenta and green all have high number of opportunities.”

The last analysis strategy is to try to come up with certain views where some attributes are all well expressed. One typical view for this strategy is shown in Fig. 23(d). Based on this view, this participant has described his findings as ‘The blue and magenta clusters have higher revenues than the green one. They both generated more leads and have lower cost per won lead. Those two factors seem to be important. Cost does not seem to play a very decisive role here.’

All participants used the STM to save important findings and they all used the STM to tell us their choices of sales strategies. Six participants simply dragged and dropped those saved views back to the trackball in order to find out the dominant dimensions on those views. Four of them used the STM to traverse between those key frames when telling us the story because they ‘like the animation’. When presenting their findings to us, all of the participants came up with similar results such as ‘The blue and the magenta teams have the highest revenue’, ‘The blue team has high number of leads and this might lead to their high revenue’ and ‘Cost per won lead needs to be low.’

All participants managed to finish all three tasks using proposed interactions after a short training. They also understood the meaning of the displays of our system well. They managed to find hidden patterns, using motion parallax to discover the third cluster and fully analyze the sales force dataset. These proves the usefulness of our system for data analysis tools

CHAPTER 5

THE 3D SHADED SHAPE REPRESENTATION

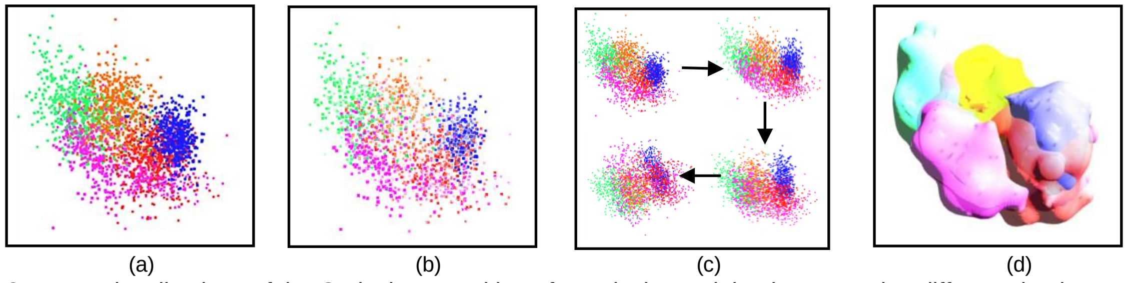


Fig. 24. Four visualizations of the Optical Recognition of Handwritten Digits data set using different depth cues. (a) Scatterplot with occlusion. (b) Scatterplot with fog function. (c) Scatterplots imitates motion parallax. (d) 3D shape representation.

Apart from cone trees and some other representation there has been a great reluctance to use 3D graphics for non-spatial data. However, recent workshops have begun exploring this (and other) issues related to 2D vs. 3D information displays. In this paper, we test the hypothesis whether transforming a data matrix into a 3D shaded surface or even a volumetric display can be more appealing to humans than a scatterplot since it makes direct use of the innate 3D scene understanding capabilities of the human visual system. In addition, we test the hypothesis whether 3D shaded displays can add a significant amount of information to the visualization of high-dimensional data, especially when enhanced with proper tools to navigate the various 3D subspaces. Our experiments suggest that mainstream users prefer shaded displays over scatterplots for visual cluster analysis tasks after receiving training for both. And further, our experiments also provide evidence that 3D displays can better communicate spatial relationships, size, and shape of clusters.

5.1 MOTIVATION

In 2009 the movie Avatar was released. It was a tremendous success and has been logged as the highest-grossing movie of all times. Avatar was so successful because it took the movie watching experience to a whole new level. Among other advances, it perfected the art of adding a third spatial dimension to the traditional two dimensions plus time. The underlying reason why this is so appealing to a human audience is because 3D is the most natural form of visual input. The human visual system is trained for 3D viewing already in early childhood and it is in these early years where an inferential chain is built that, fed by the retinal stage, takes in surrounding information and outputs 3D shape relations, aided by 3D models of the world already stored in the brain.

In the area of inform visualization researchers have long realized the benefits of incorporating all three spatial dimensions into the information display. This of course is natural since the data in these fields typically have a spatial reference. This is not the case for information visualization where data are usually from non-spatial sources and have no direct relation to 3D space. Henceforth, 3D methods have not been widely applied in information visualization. Scatterplots, parallel coordinates, tree maps, and the like all embody 2D graphics. One of the few exceptions are cone-trees [75]. Others have used 3D shading to accentuate boundaries of 2D shapes, such as cushion tree maps [97]. Finally, some researchers [85][69] have exploited motion parallax which is 2D plus time to disambiguate distance relations in scatterplots displays. We are not aware of research that has used fully 3D methods for the depiction of these types of point clouds.

We define 3D visualizations as renderings that include depth cues, such as occlusion, shading, perspective distortion, shadows, and so on. As discussed, these are all phenomena that occur in the daily life of every normal person, and it is this audience we would like to get interested and engaged in the visualization of complex data. Adding 3D shading effects to bar and pie charts, as many plotting programs do, seems to be a step into the right direction. While it is arguably a rather small step, it does make data visualizations more appealing to the general user, as 1,000s of business graphics can readily attest to.

Interesting to our mission is a recent study conducted in the Smithsonian Museum in Washington DC [115] where visitors were asked to rate certain abstract 3D shapes for aesthetics. The study suggested that curved 3D shapes are more attractive to humans, as opposed to non-curved ones, offering them more aesthetic pleasure. In fact, the 3D shapes our renderer produces are strikingly similar to these curved museum pieces, although they are shaped by data and not by artists. In our work, shape is the high-dimensional manifold covering a point distribution or cluster. Since graphics displays can only render in 3D, we project these high-dimensional shapes into 3D, and provide an interactive interface by which users can navigate the high-dimensional space, control the 3D projection operations, and so manage the large space of possible 3D projective shapes due to data subspaces, seamlessly transitioning among them.

The high-dimensional navigation and exploration uses our existing subspace exploration tool - the Subspace Voyager [94]. However, instead of operating on the data points directly, we now use the Marching Cubes algorithm [60] to extract the 3D shapes of the density fields. Multi-layer semi-transparent rendering can then visually reveal hidden structures as well as phenomena associated with varying density of the data, such as skew and outliers.

To illustrate what has been stated above, Fig. 24 shows a comparison of 2D vs. 3D, using the Optical Recognition of Handwritten Digits dataset [116]. This dataset has the normalized bitmaps of the ‘0’-‘9’ handwritten digits. We only use 5 clusters out of the 10 (‘0’-‘4’) for ease of comparison. Fig. 24(a) is a scatterplot with occlusion implemented for depth perception (points in the back are blocked by points in the front) viewed from an optimized angle where most of the clusters are separated. Fig. 24(b) is the same scatterplot viewed from the same angle but with a fog function applied (points in the back are dimed by an invisible fog). Fig. 24(c) uses four frames to imitate motion parallax where the scatterplot from Fig. 24(a) is tilted upward. Fig. 24(d) is our 3D shape representation with the shapes of the five clusters extracted. We observe that with our method (Fig. 24(d)), the structures and relations of the five clusters can be clearly discerned. The blue cluster and the red cluster are on the right and the red cluster surrounds the blue cluster. The purple and the red cluster are both close in the front while the yellow cluster is in the back. Most of the green cluster is well separated from the rest of the clusters but part of it is on top of the purple one. We also find that the third dimension, in combination with the shaded display, allows users to peek around cluster shapes, mitigating the point overlaps that exist in the scatterplot projection (Fig. 24(a) and Fig. 24(b)).

One may argue that these relations can also be observed by other visualizations, especially those that utilize motion parallax. But these methods require interaction to generate the effect, while shaded displays invoke motion parallax by sheer 3D cognition.

As mentioned above, information visualization deals mostly with non-spatial data which can be multivariate. So one might say that adding a third dimension for the visualization of these data only offers insignificant gains when the number of variables is on the order of 10s, 100s, or even 1,000s. While this is a valid argument, it is not the reason why we advocate for 3D graphics. Rather, we advocate for it since, as the wide popularity of 3D movies and 3D shaded bar and pie charts readily show, 3D graphics is more appealing and potentially more engaging to general users than plain 2D graphics. And furthermore, due to the innate 3D reasoning capabilities of the human visual system, 3D renderings can also potentially facilitate better data understanding. We successfully tested both of these hypotheses for the research in this dissertation.

5.2 SYSTEM OVERVIEW

Our 3D shaded shape representation (Fig. 25) is tightly integrated in our Subspace Voyager [94] which is described in detail in chapter 3. Fig. 25(a) shows this interface with one of the shape displays we propose in this paper -- the corresponding point display is shown in Fig. 25(b). The circle containing the shape (or the points) in the SE is the boundary of the trackball interface. At its periphery are the current directions of the projected dimension axis vectors shown as dimension labels. All of the original navigation controls are still available, and new controls (Fig. 25(c)) related to shape generations have been added to the interface. The two checkboxes are used to control if the shape representation, instead of regular scatterplot should be displayed, and with/without the original points. The sliders are used for controlling the generated shapes, such as the transparency, number of layers and level of detail.

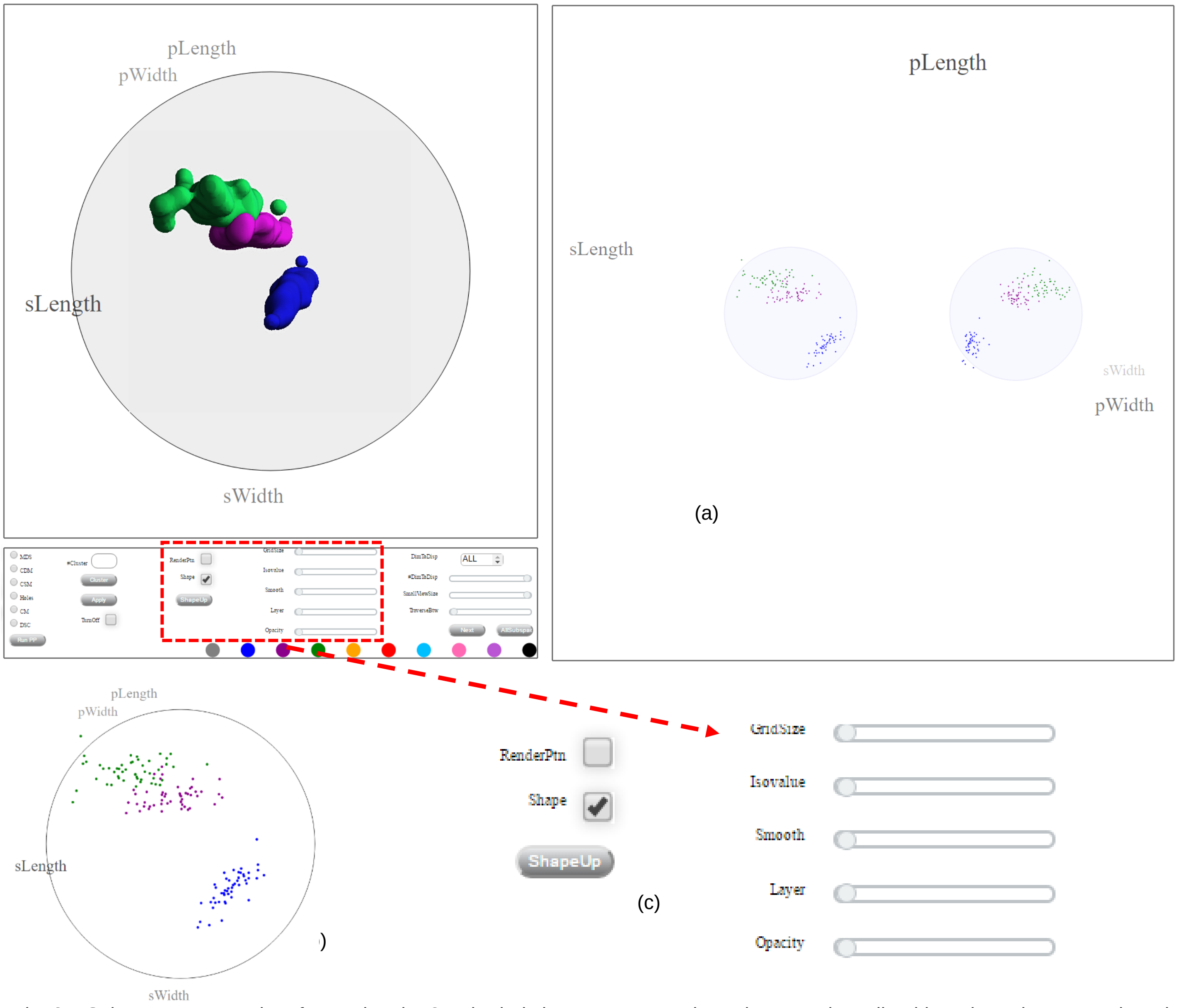


Fig. 25. Subspcay Voyager interface using the 3D shaded shape representation The controls outlined in red are those new in order to modify the shapes.

5.3 METHODS

The first step in building our shaded shape representation (SSR) is to build a 3D space where the high-D data is projected into. We then run the Marching Cubes algorithm [60] to find the polygon mesh enclosing the projected points. It is similar to the routine proposed in [63] but their work still strives for a 2D representation. We shall now describe how this works along with the various rendering modes our system provides.

5.3.1 Creating the 3D space

The creation of the polygon mesh requires 3D points and for data sets with more than three dimensions, we need to create a 3×N projection matrix P. We will refer to the three N dimensional vectors as *Projection Plane Axis (PPA).* We have two options for the first two of the vectors in P: (1) using randomized projections [6] and obtain the orthogonal PPA x-axis and y-axis pair, or (2) perform Principal Component Analysis (PCA) [40] and use the two most significant PCs. In both cases we require a third orthogonal axis – the PPA z-axis. To select an initial vector for the PPA z-axis we can either use the third vector obtained from randomized projection or PCA, or let the user manually assign one by pointing in the Subspace Trail Map. If this PPA z-axis is not orthonormal to both PPA x-axis and PPA y-axis, we perform Gram-Schmidt orthonormalization process [31]. With the projection matrix P in place, we multiply each N-D point vector V^{ND} by P to obtain the 3D points $V^{3D}=M\cdot V^{ND}$.

5.3.2 Generating the Shapes

We run the Marching Cubes (MC) Algorithm [60] on the 3D points to find their shapes. MC operates on 3D spatial data interactively and builds triangle mesh based isosurfaces. One can imagine a virtual 3D grid subdividing the space with all points residing on its vertices. Some vertices may have more points indicating high isovalues while others may have no points at all. If the isovalues of all 8 vertices of a cube are smaller or bigger than the user defined threshold, the isosurface does not intersect this cube. Otherwise, the isosurface crosses this cube in the form of a small mesh of triangles. Because each vertex can either be above or below the isosurface, there are 2^8 different configurations for the cubes (In fact, with mirroring and rotating, the 256 possibilities can be reduced to 15 different unique cases). MC orders all 8 vertices from 0 – 7, and based on the values they have (1 for higher than the threshold and 0 for lower), an 8-bit binary number is formed. This number can be used as an index to find the entry in a lookup table that determines which edges of the cube the triangle intersects. Aggregating all triangles produces the final isosurface.

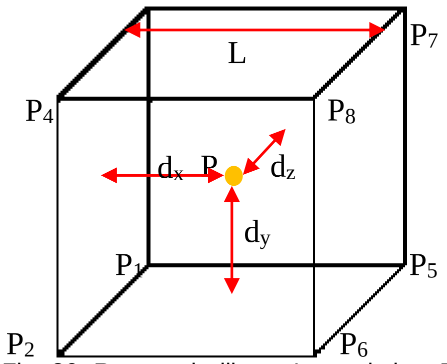


Fig. 26. Reversed trilinear Interpolation. The values of the 8 vertices are approximated based on the distances between the faces of the cube and the actual point.

One small caveat is that we are mostly dealing with non-spatial data and these 3D points are irregularly sampled and are not perfectly aligned with the vertices of the grid. We solve this problem by *splatting* the points into a regular grid using reversed trilinear interpolation. Suppose the length of the edges of the cubes is L and the distances of a data point to the left, bottom and back faces of the cube which it resides in are d_x, d_y and d_z, respectively (see Fig. 26). Let $w_x = d_x/L$, $w_y = d_y/L$ and $w_z = d_z/L$. The values on the 8 vertices are then approximated using the following formula:

$$
\begin{gathered}
V_{p_1} = w_x * w_y * w_z \\
V_{p_2} = w_x * w_y * (1 - w_z) \\
V_{p_3} = w_x * \left(1 - w_y\right) * w_z \\
V_{p_4} = w_x * (1 - w_y) * (1 - w_z) \\
V_{p_5} = (1 - w_x) * w_y * w_z \\
V_{p_6} = (1 - w_x) * w_y * (1 - w_z) \\
V_{p_7} = (1 - w_x) * \left(1 - w_y\right) * w_z \\
V_{p_8} = (1 - w_x) * (1 - w_y) * (1 - w_z)
\end{gathered}
$$

After this step, we also apply a 3D Gaussian filter on the gird to smooth out the points since otherwise the final shapes usually appear jagged. We use a 9 × 9 × 9 Gaussian filter and instead of applying it on the points for three nested iterations, we separating the filter into X, Y and Z direction to speed up this process. We offer point light and ambient light in our system. The point light acts like a light bulb and either creates bright areas or casts shadows on the objects depending on their positions and structures. Conversely, the ambient light adds uniform shading to all objects in the scene, regardless of their properties.

We use a sales campaign data set as an illustrative example in Fig. 27. Our data set consists of 900 points (one per salesperson) and 10 attributes parameterizing the basic corporate sales pipeline. There are 3 clusters. Fig. 27(a) shows the generated shapes and their corresponding 3D points (top right). e use the same color for the same cluster in both views. We also take another view of the data and generate its shapes in Fig. 27(b). We already see the resemblances in the shape representations with their original scatterplots, such as the relative positions and the silhouettes of the 3 clusters. We will demonstrate the correctness of our representation later.

There are three parameters that can be adjusted here – the number of cubes that the grid has, the number of times the box filter is run, and the isovalue. They can be adjusted using the GridSize, Smooth and IsoValue sliders in the control panel (Fig. 25(c)), respectively.

We find that with more cubes in the grid, the details of the shape are better depicted (Fig. 28). When the grid size is 30×30×30, the shapes grow very large, fill almost the entire space, and look like three shapeless blobs (Fig. 28(a)). But when the grid size is 130×130×130, small lumps

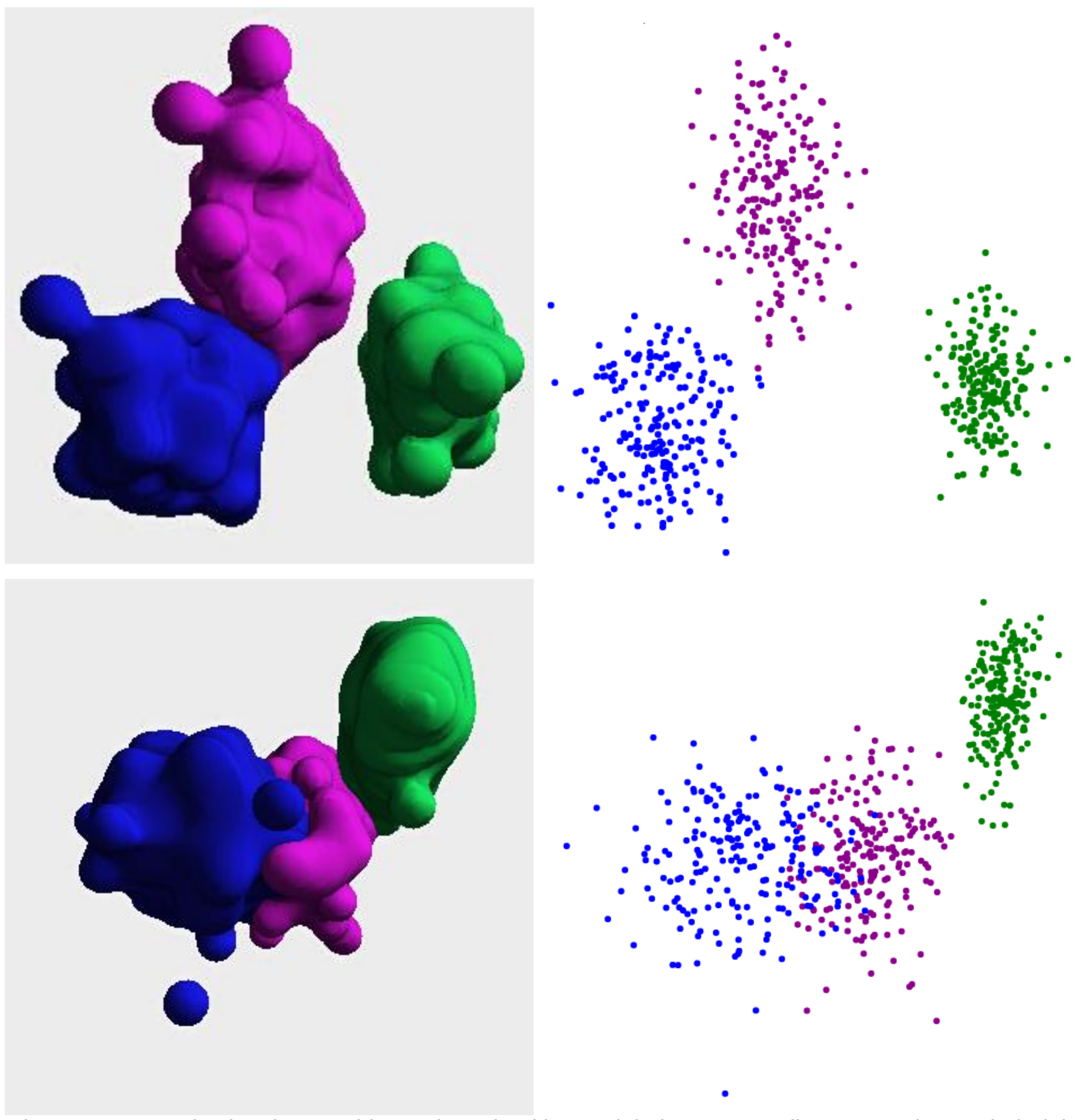


Fig. 27. Shapes generated using the Marching Cubes algorithm and their corresponding scatterplots to their right.

formed by points on the boundary are shown (Fig. 28(c)). A grid size of 80×80×80 seems best here.

The number of smoothing iterations has the opposite effect (Fig. 29) – the greater the number of iterations is, the less details the shapes have. The smoothing stage works like the blurring filter in image processing; when applied, details (in our case, sparse points or outliers) are reduced. Without smoothing, individual points instead of shapes are generated (Fig. 29(a)); with an increased amount of smoothing, the lumps on the isosurfaces disappear (Fig. 29(c)).

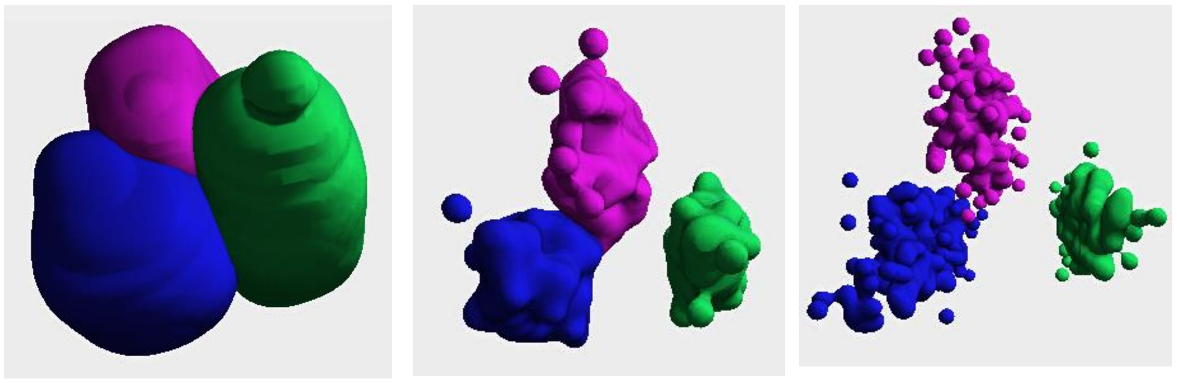


Fig. 28. The effect of grid resolution on the generated shapes: Left: 30×30×30, middle: 80×80×80, right: 130×130×130

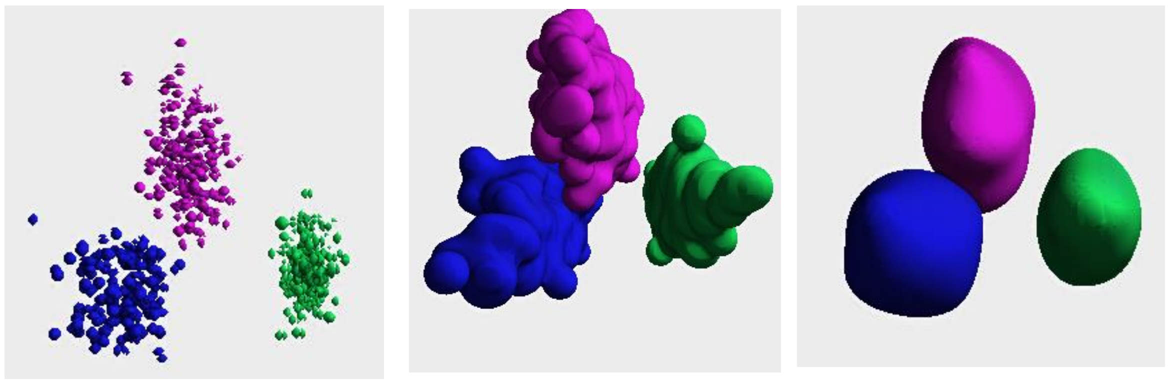


Fig. 29. The effect of number of smoothing iterations on the generated shapes. Left: No smoothing. Middle: 2 iterations. Right: 10 iterations

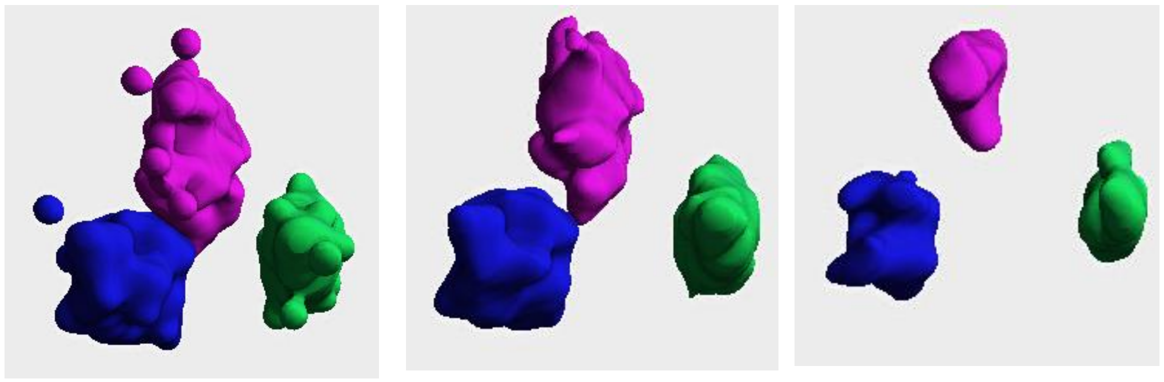


Fig. 30. The effect on the final shapes when using different isovalues. Left: Isovalue is 0.00075. Middle: Isovalue is 0.00375. Rigth: Isovalue is 0.01

Finally, the isovalue controls which portions of the points are rendered – with a higher isovalue the denser part of the points would be used to generate the shapes (Fig. 30(c)). In contrast, all points would be enclosed in the shapes if the isovalue is low (Fig. 30(a)).

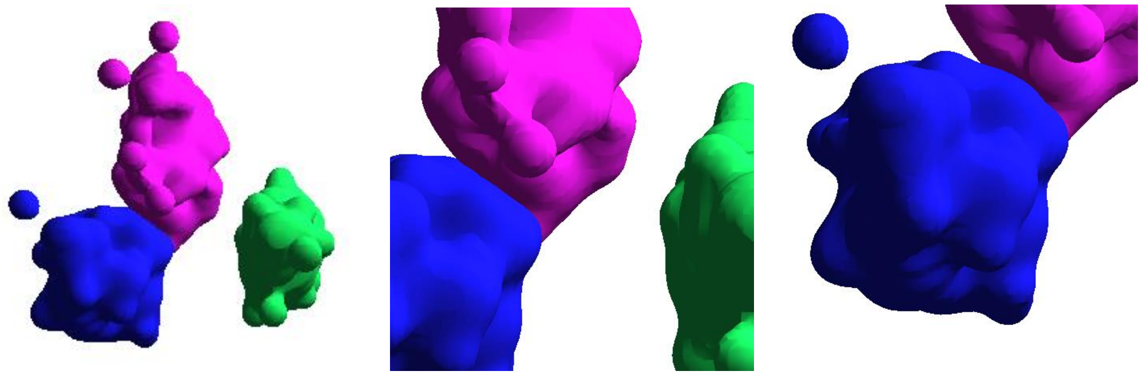


Fig. 31. Zooming and panning. (a) The shapes are zoomed out. (b) The shapes are zoomed in. (c) The shapes are zoomed in and panned to the blue cluster

5.3.3 Manipulating the Shapes

We provide the user with the flexibility to render the generated shapes as they like. The following adjustments are supported:

- **Zooming and panning.** Users can apply the scroll wheel to zoom in/out the shapes (Fig. 8). This allows them to inspect the shapes closely or attain an overall view of them. Users can also pan to the desired areas of the shapes by moving the mouse while the right mouse button is pressed.

- **Transparency**. Users can change the transparency of the shapes, and shapes with higher transparency (lower opacity) reveal the structures that are otherwise hidden behind them. Fig. 9(a-c) show the same shapes rendered with different opacities. Both Fig. 9(a), which renders the purple object at half opacity and Fig. 9(b), which renders it at 70% opacity allow the user to see the fairly large blue cluster that is behind the purple one. Conversely, Fig. 9(c) where the purple object is fully opaque yields no information on how the large the blue cluster really is.

- **Multi-layer**. In Computed Tomography (CT) scan, high density areas are reconstructed using solid surfaces while low density areas are depicted with low opacities. This easily separates body parts with different densities and makes diagnosis less complicated. We believe that if we could create our shaded shapes using a similar method, it would also help users to discern the inner structure of the data. As discussed in section 4.2.1, MC with different isovalues renders different parts of the points – denser parts would only be rendered if larger isovalues are applied. We can rely on this property and use different isovalues at the same time. Fig. 33(a) is one such example. We run MC using three isovalues and the final results has three layers. The outermost layer is almost transparent and it includes almost all the points while the innermost

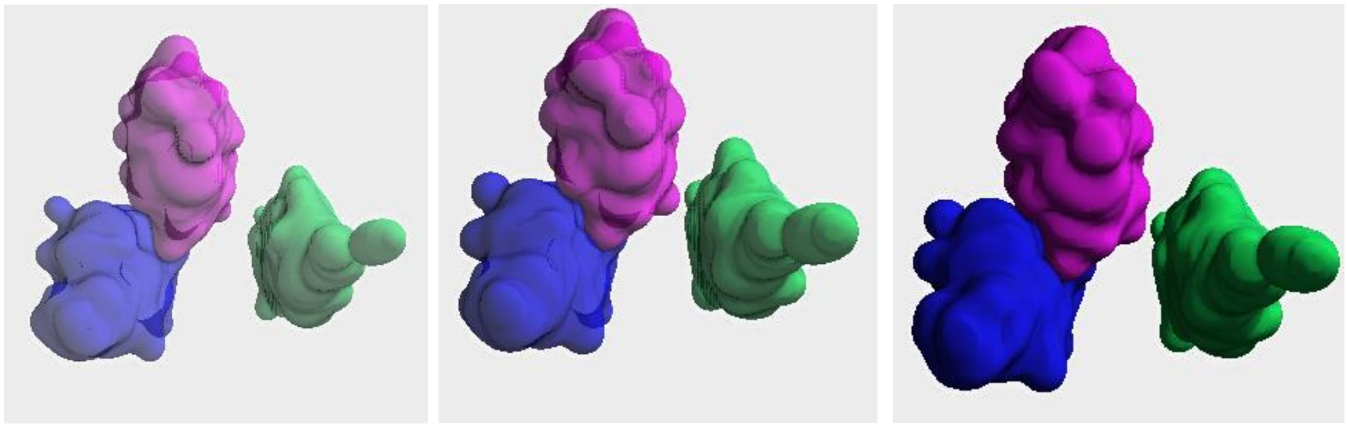


Fig. 32. Different transparency. Left: Opacity = 0.5. Middle: Opacity = 0.7. Right: Opacity = 1.

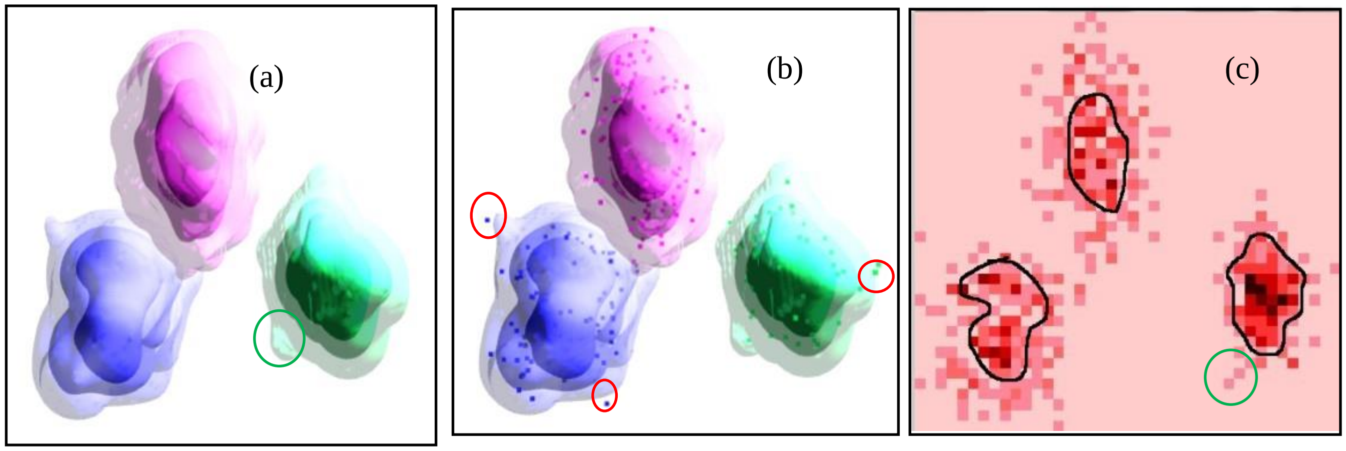


Fig. 33. The Multi-layer mode and the Points within Shapes mode. (a) Multi-layer mode. (b) Render points within shapes. (c) The corresponding heat map. We can see that the dense parts in (c) map to the innermost layers in (a) and (b). Certain distributions of the dataset (green circle in (c)) are also preserved (green circle in (a)). The left out points circled in red in (b) are far from the majority of the points and can be considered outliers.

layer is opaque and only points in denser areas are included. Users can clearly tell the dense parts apart from the other parts and so gain insight into the data set.

- **Points within shapes**. We also allow users to render the shapes and the points at the same time. Fig. 33(b) gives such an example.

The transparency and the number of layers of the shapes can be controlled by adjusting the *Opacity* and *Layers* sliders in the control panel. The points within the shapes can be turned on and off by the *RenderPtn* checkboxes in the control panel.

5.3.4 Proof of Correctness of our Shape Representation

With the proposed shape representation being a new paradigm for scatterplot rendering it is important to prove its correctness.

Let us first consider Fig. 33(b) where all points are rendered together with the shapes. There we notice that all points, except for the three circled ones, are enclosed by the shapes. The curvatures of the shapes also fit well with the edge points. The three left out points are far away from the rest of the data, and because of the Gaussian filter we applied, the eight vertices of the cubes they reside in have little weight. They still influence the final shape (see the point in the blue lump on the outermost layer that is in the red circle), but MC did not include them in the generated isosurfaces based on the isovalue we chose. We wish to point out that because MC is isovalue sensitive, these three points can be included in the shapes by assigning a lower isovalue or by using fewer rounds of box filtering.

To give a more quantitative analysis, we also generated a heat map based on the 3D density of the original non-splatted points (see Fig. 33(c)). By comparing Fig. 33(a, c) we can see that the shapes preserve the original distribution of the points well. For example, the tail of the green cluster in Fig. 33(a) circled in green matches the parts in Fig. 33(c) circled using the same color. We also drew the contours enclosing the dense areas of the three clusters on the heat map (black lines), and we can see that they roughly match the silhouettes of the innermost layers in Fig. 33(a).

It should be mentioned that since we generate the shapes based on the splatted and smoothed points, sometimes only the overall trend and not every detail of the points is retained. If we had used a grid with higher resolution and fewer smoothing iterations, more details would have been maintained but at the cost of higher storage and rendering. In addition, sample points in data science are usually always associated with some inherent uncertainty. One might design a grid size and blurring filer that would be tuned to this uncertainty.

5.4 CASE STUDIES

We had set out to design our 3D framework with the intent that it might (1) help clarify certain relationships and (2) help engage users to move along in their data exploration task, providing them with interesting options to explore, akin to game play. For the former one might have a conventional scatterplot display augmented with a ‘ShapeUp’ button that would provide an instant 3D view of the same scene. As noted, both of these intents were motivated by the human visual system’s natural penchant for 3D displays. In the following we provide two case studies to bring home these points, complemented by the results of a web-based user study we conducted.

5.4.1 Using the Shape Displays to Clarify and Engage

We use Fisher’s Iris data set [114] as a first example. The Iris flower data set has three classes with 50 instances in each of them. Each class belongs to a different type of iris. This dataset has five attributes – sepal length, sepal width, petal length, petal width, and the class label. Data analysts often use a SPLOM to exploit it. We use the first four attributes to generate half of the scatterplots in the SPLOM and place their corresponding SSRs to their left (see Fig. 34).

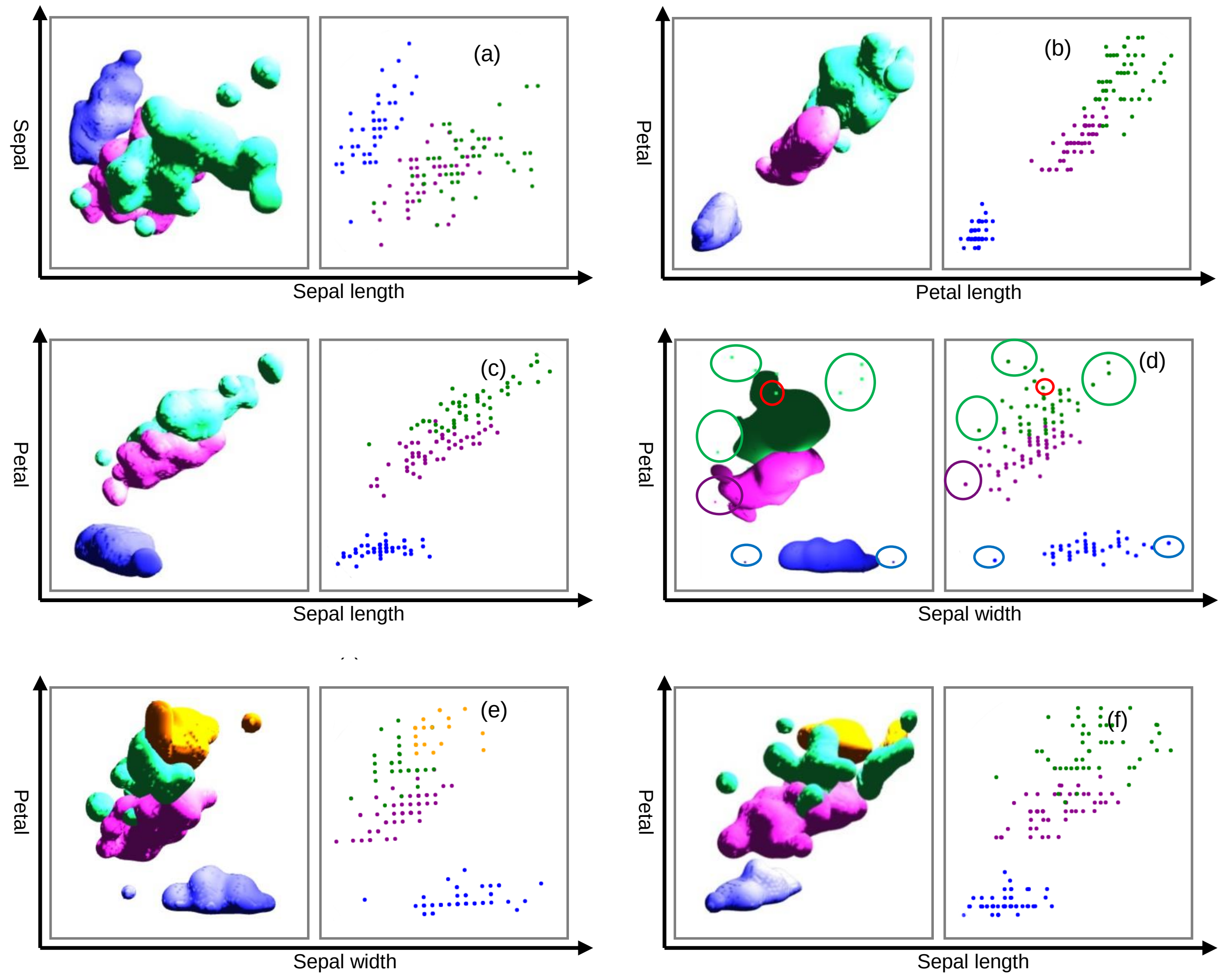


Fig. 34. Comparisons between scatterplots and our shaded shape representation using the Iris dataset. (a) The green and purple clusters are not separable in the scatterplot but our SSR suggests the possibility to use a plane parallel to the screen to separate them. (b)(c) From the scatterplots, only the 2D distribution of the data can be seen but our SSR allows users to observe the trend along the third dimension. (d) Outlier detection. Our SSR is able to detect outliers that are far away from the majority of points along the third dimension (red circle). (e)(f) Brushing and linking. A portion of the green cluster is painted yellow in (e), all points belonging to that portion form a new cluster for shape generation in (f).

Essentially, each view on the left could be the 3D renderings that would result when hitting the aforementioned ‘Clarify’ button when only the 2D scatterplot on the right is seen.

Let us assume a plant biologist, Tim, views the scatterplot display in Fig. 34(a) on the right which is a rather complex one. While he can easily discern that the blue cluster of flowers is separate, he cannot tell for sure for the green and purple clusters. So he hits the ‘ShapeUp’ button which produces the 3D view to the left in Fig. 34(a). He can now clearly see that the green cluster is in front of the purple one and a plane parallel to the screen seems to be able to separate them. This gives Tim good guidance on how to find a proper view where all three clusters can be separated. Since the current view is Sepal width over Sepal length, he now knows that the other attributes, Petal width and Petal length, or both, are also involved in the classification.

Tim moves on and inspects a 2D scatterplot of Sepal width and Petal length (Fig. 34(d), right). Looking at this display he suspects that there might be some flowers that are different from the main stream – outliers. To get more insight he hits the magic ‘ShapeUp’ button and the system produces the 3D display on the left. Now he clearly sees that the points he suspected to be outliers (circled in green and blue) are indeed outliers. But he also discovers an outlier he did not suspect – the one circled in red. This point is indistinguishable from the others in the 2D scatterplot, but pops out easily in the associated 3D SSR. While Tim could have possible found this rare flower in a different 2D scatterplot, the 3D SSR is more direct and does not require a context switch to a different set of dimensions. Rather, it uses the same dimensions than the 2D scatterplot, just using shading to disambiguate it.

Next, Tim calls up the 2D scatterplot that visualizes Petal length vs. Petal width (Fig. 34(b), right). He sees three lean shapes that spread along the diagonal, suggesting a strong correlation for the green and purple classes. The blue class, on the other hand, does not seem to have a trend. Tim thinks that this is interesting but he is not certain what 2D scatterplot to look at next. He presses the ‘Clarify’ button (producing the view on the left) and sees that all three classes seems to protrude into the depth direction, almost inviting him to follow to see what this trend is all about. He knows that this third dimension has to be one of the remaining ones, or a combination of them. So he gets curious again and exchanges Petal with by Sepal length (see Fig. 34(c)) to learn more about his data. And the exploration continues.

An important activity in interactive cluster analysis is brushing and linking. Tim decides that he would like to break away part of the green class into a separate class since he thinks that it an important subset of flowers. Tim prefers to do this in 3D since he used to painting 3D objects. He picks the yellow color and brushes on a portion of the green surface. Our system automatically assigns the tagged surface points, as well as points underneath, to a different cluster. This produces Fig. 34(e), left. Next, Tim uses the trackball interface to go to a different projection (Fig. 34(f), left) where new shapes are generated according to the new axis dimensions. Tim observes that the tagged yellow cluster is now rotated to the back. This gives Tim an overall idea of the dynamics of what he thinks are the interesting parts of the data.

5.4.2 DISCOVERING HIDDEN STRUCTURES

We present a somewhat contrived, almost malicious, dataset that we constructed to illustrate that our SSR can make structures visible that equivalent 3×3 SPLOM cannot. Such an example readily extends to higher dimensions – we just chose 3D for ease of illustration.

We used our SketchPadND [95] to generate a test data set. The data set is a box shaped point cloud with four sides closed (except for the front and the bottom sides, Fig. 35). There are three

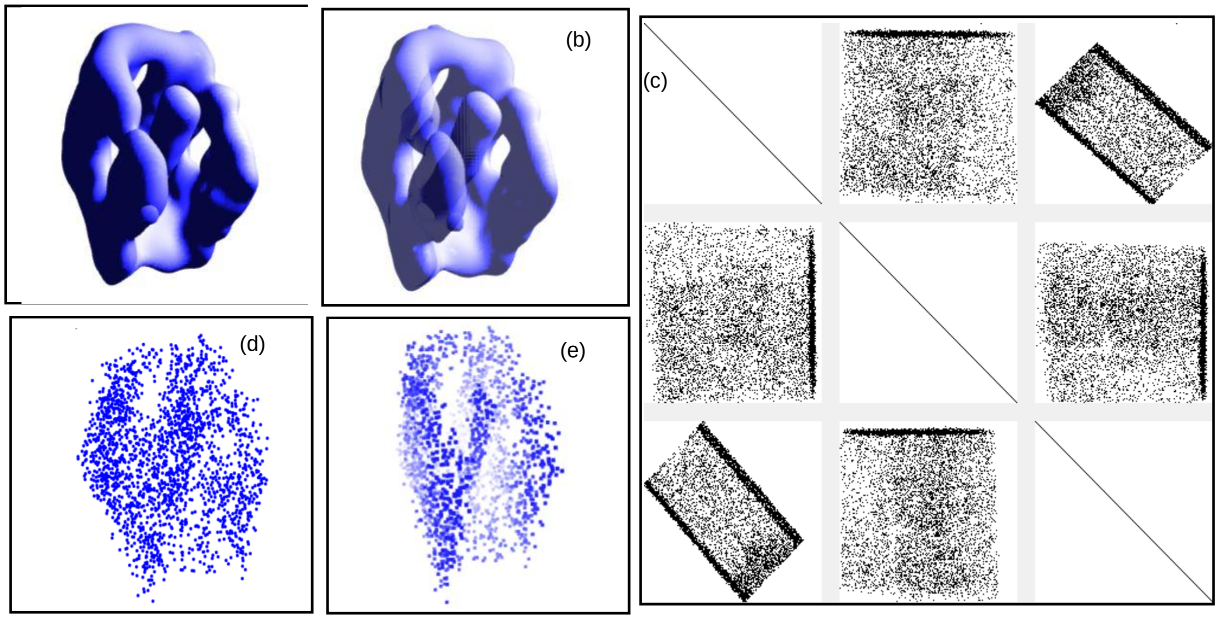


Fig. 35. Our shaded shape representation helps discern hidden structures. (a) Shaded shape representation with opacity set to 1.0. (b) Shaded shape representation with opacity set to 0.7. (c) Scatterplot Matrix. (d) Points viewed from the same angle. (e) Points viewed from the same angle with fog function applied.

holes on the top, left and right sides and one solid stick in the middle of the box. In Fig. 35(a), we can clearly see the structure of the data. The stick is pointing out in the middle and the three holes are also obvious. If we change the opacity of the shape from 1.0 to 0.7 (Fig. 35(b)), more attributes can be seen. For example, we can now see the entire hole on the back and tell where the stick ends. We compare our SSR with an SPLOM in Fig. 35(c). Since the stick is perfectly aligned with the holes in all axis-aligned views, we cannot determine the structure simply using SPLOM alone. The only thing we can discern is that there are some denser planes (the sides of the box), but we would never picture holes on those planes or the existence of the inner stick. If we look at the points from the same angle as the shaded shape (Fig. 35(d)), we do notice two holes (top and right), but because of points overlapping, neither the left hole nor the stick can be observed. Finally, we add invisible fog in the scene (Fig. 35(e)), and the tip of the stick becomes visible. The left hole is a bit clearer than Fig. 35(d) but still not quite obvious. However, the emergence of these two structures comes at a cost: the other two holes are now blurred out. The fog dimmed the points in the back too much and the rear part of the shape seems hollow now.

Admittedly, with a movie showing this object rotating around, all three holes and the stick can be observed. However, a movie takes more computing resources to generate and it also costs more time for users to reason with a movie. In fact, with a movie clip, viewers typically forget what they have seen before and cannot connect the dots. This is not a problem for our SSR because users only need to look at one still image.

5.5 USER STUDY

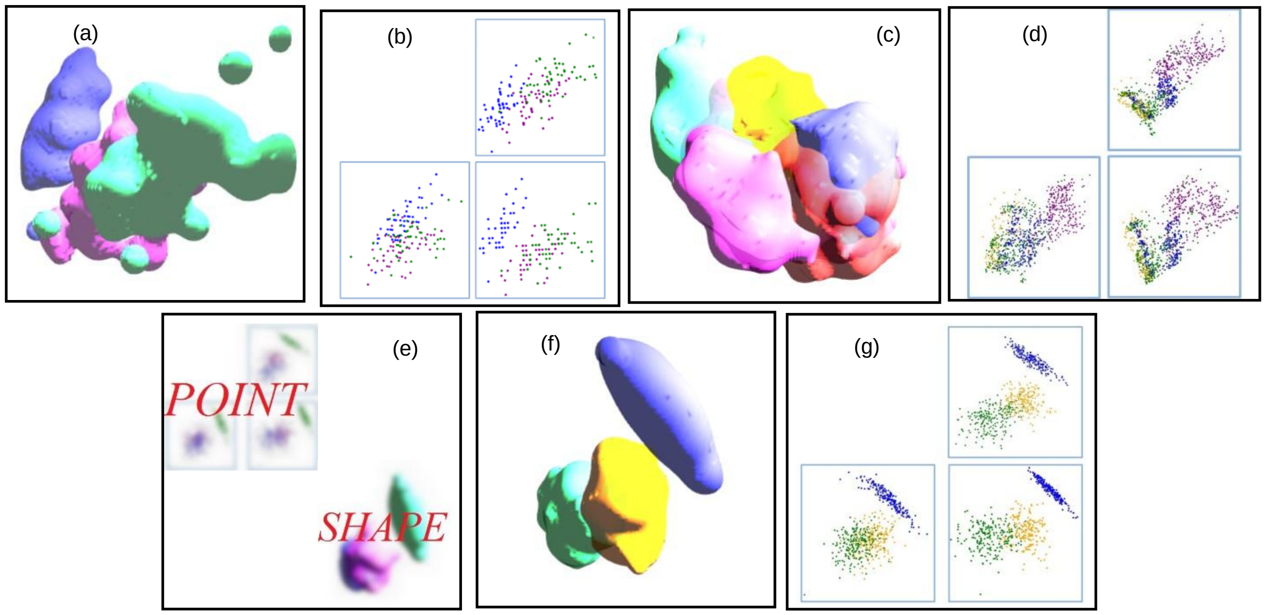


Fig. 36. Data sets used for our user study. (a) – (g). The different SSR and the scatterplot representations used in the user study.

Sections 4.4 has shown that our SSR can nicely complement and possibly substitute a traditional 2D scatterplot display or SPLOM for analysis. We also hinted on the possibility that a 3D display might also be more engaging and appealing to human users. In this section we have set out to formally proof all this via a targeted user study. Specifically, we want to know, do mainstream users like the SSR? Is it more appealing than point-based displays? Is it more engaging? What about accuracy? To test these questions, we conducted a user study on Amazon Mechanical Turk.

5.5.1 THE SETUP

Our study started with the explanation of what a cluster is and showed the participants the two different ways to represent different clusters – SSR and scatterplots (Fig. 36(a)(b)). For the latter, we used a combination of three scatterplots because we wanted to give the participants more depth information. In the bottom right corner of Fig. 36(b), the data set is viewed from the same angle as in Fig. 36(a). In the bottom left corner, the viewing window has been rotated toward the left relative to Fig. 36(a), whereas in the top right corner, it has been rotated upward. The three scatterplots provide the participants with motion stimulus and the relative locations of the three clusters can be derived. In Fig. 36(b), because we rotate the bottom right scatterplot along its center to the top (top right scatterplot), the points in the back were moved toward the bottom (the blue cluster) and points in the front were moved toward the top (purple and green cluster). Since the green cluster is in front of the purple cluster and further away from the rotation center, it has larger displacement compared to the purple cluster in the same scatterplot. Similar reasoning can be made for the upward tilting.

5.5.2 Study Design and Tasks

Following these instructions, we asked the participants three sets of questions. The first set featured a visualization using our SSR (Fig. 36(c), the Optical Recognition of Handwritten Digits data set [116]) and the second set featured a visualization using the scatterplots (Fig. 36(d), the Image Segmentation data set [117] where we chose 4 clusters out of the 7). Hence, each participant saw both representations, had to use them for problem solving, and so was familiar how each operates. We used different datasets for both to avoid learning effects.

Next, for answering the last set of questions, each participant had a choice between the SSR (Fig. 36(f)) and the scatterplot (Fig. 36(g)) to answer them. Both used the Campaign dataset since learning was not an issue here. To select the representation of choice they could click on one of the pictures in Fig. 36(e). The 'SHAPE' labeled picture represents our SSR, while the 'POINTS' labeled picture represents the scatterplots.

Each set contained three questions. The first questions were always "How many clusters are there?" the second questions were "Which two clusters are better separated?" and the third ones were "Which cluster is in front?" All of the questions were binary choice. For example, for the first set of questions (Fig. 36(c)), we offered them "3" and "4" for the first question, "Blue & Red" and "Blue & Green" for the second one and "Purple" and "Yellow" for the third one.

Since we make reference to colors, we conducted the Ishihara color blindness test [118] before the real user study and only those participants who had passed this test could proceed to the survey.

Specifically, our three hypotheses were:

H1: the participants would demonstrate equal performances for the first questions related to the number of clusters using both method since they do not make key use of the depth cue;

H2: the participants would perform better for the second and third questions when using our SSR since it's easier to determine the spatial relation; and

H3: a majority of the participants would choose our SSR over scatterplots because it is more appealing and natural to them.

5.5.3 Launch, Analysis and Outcome

Table 2. Results for our user study. The numbers before the slash are the counts of correct answers while the numbers after are the total responses.

	SSR	Scatterplot
# of clusters	72/74	42/46
Separability	65/74	33/46
In the front	62/74	24/46

Table 3 Chi Square statistics and p-values for our user study. This confirms our three hypotheses. Participants performed equally well in counting the number of clusters using both representations but our SSR offers higher accuracy in separating clusters and figuring out spatial relations between clusters..

	Chi-Square	p-value
# of clusters	2.1448	0.143049
Separability	4.9103	0.026698
In the front	13.9585	0.000187

We posted 40 assignments for our HIT (Human Intelligence Task) on Amazon Mechanical Turk. We chose Amazon Mechanical Turk in order to get a diverse pool of participants. For the last set of questions, even though it uses the same data set and the same questions for both representations, the number of participants that answered them for each was different. We therefore combined these responses and counted the number of correct and incorrect answers to the three types of questions (cluster number, cluster separability and which cluster is in the front). All 40 participants answered the three questions in set 1 and 2, 34 chose our SSR to answer the questions in set 3, while 6 chose to use the scatterplot representation. This yields 74 (40 + 34) responses for each type of answers using our SSR and 46 (40 + 6) answers for each type of answers using the scatterplot representation. The aggregated results are shown in Table 2. The numbers before the slash are the counts of correct answers while the numbers after the slash are the total responses.

To test the first two hypotheses, we performed the Chi Square test [99] to get the p-values [20]. The Chi Square statistic indicates how well the observed values fit with the expected values based on a certain hypothesis. The p-value is a relative standard to accept or reject this hypothesis ($p<0.05$ means the result is significant) and can be looked up using the Chi Square statistic.

The calculated Chi Square statistics and p-values for the three types of questions are shown in Table 3. We can see that the p-value for testing if there is a difference in the performances of counting the number of clusters using the two different representations is not significant

(0.143049 > 0.05). This insignificance confirms H1. Since this task is simply about counting the number of different colors, it should be easy for both representations. The second p-value (0.026698 < 0.05) indicates that it is easier to figure out which clusters are better separated in our SSR. The last p-value 0.000187 is much smaller than 0.05, indicating that our SSR is much better at showing spatial relations compared to scatterplots. This confirms H2.

Out of the 40 participants, 34 chose to use our SSR to answer the last set of questions and only 6 of them chose to use scatterplots. We computed the confidence interval (11.07%) for this observed proportion (85%) of participants that preferred our SSR to the scatterplot representation. This result indicates that we can be 95% certain that the true population proportion of people that prefer our SSR falls within the range of 73.93% to 96.07%. This suggests that H3 is also true. The majority of the participants might find our SSR more informative and easier to interpret than scatterplots and that is why they chose our SSR. Another possible explanation is that they simply found our SSR more appealing and engaging.

CHAPTER 6

THE SCATTERPLOT BASED DATA GENERATION TOOL

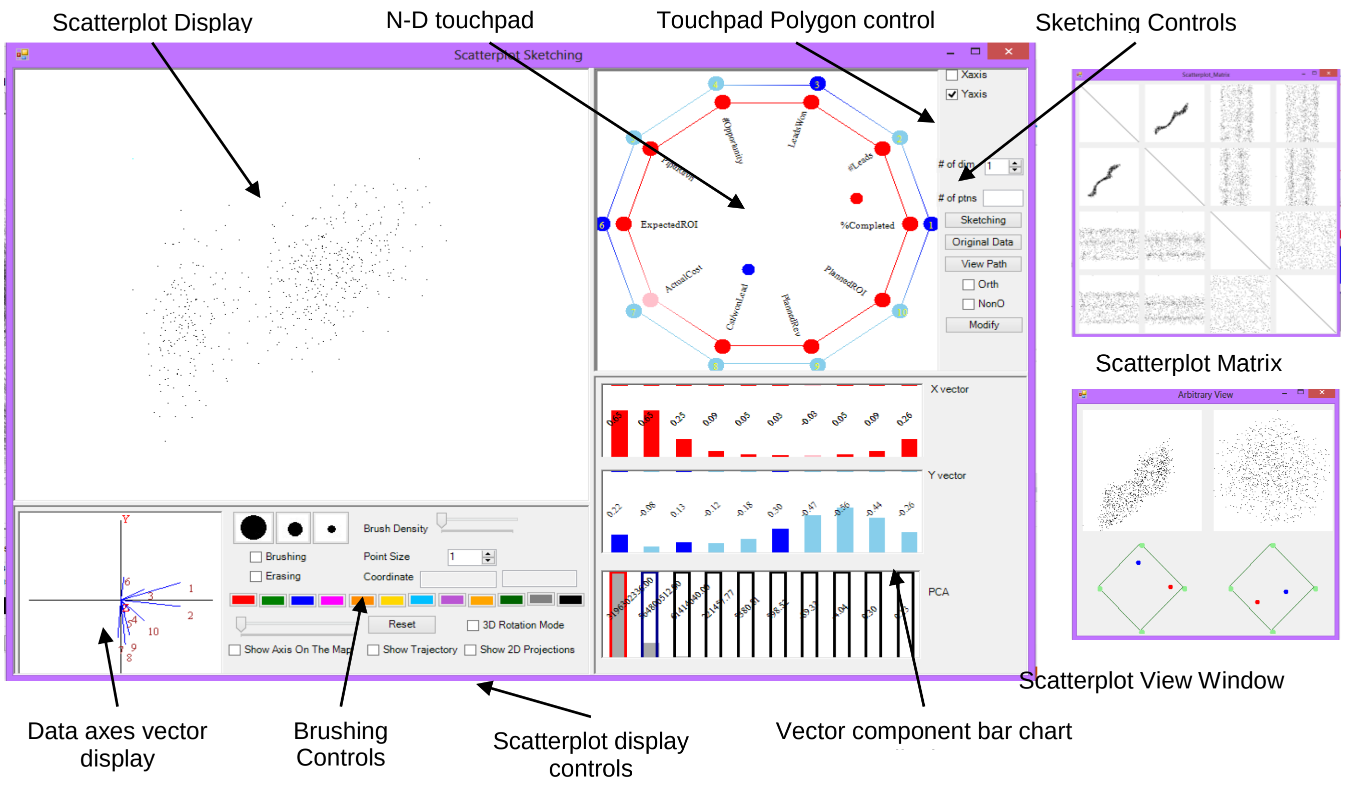


Fig. 37. Scatterplot Sketching interface

High-dimensional data visualization has been a very popular research topic recently and there are a large number of tools and frameworks that have been designed for this purpose. In order to test those tools, researchers always need datasets with certain desired features. However, those features are not always present in real world dataset, or only partially available. Here we propose the a scatterplot based WYDIWYG (What You Draw Is What You Get) data generation tool. It allows users to generate high-dimensional data in the same interface they also use for visualization. This provides for an immersive and direct data generation activity, and furthermore it also enables users to interactively edit and clean existing high-dimensional data from possible artifacts. Our tool is based on a relatively new framework using an N-D polygon to navigate in high-dimensional space, and it embraces the idea of sculpting. Users can carve data at arbitrary orientations and refine them wherever necessary. This guarantees that the data generated is truly high-dimensional.

6.1 MOTIVATION

High dimensional data is immersive in our life nowadays and it's useful in many applications and domains. In order to analyse it, researchers has designed a larger number of techniques and tools. These new algorithms and software all need to be formerly tested using datasets with specific features. However, those datasets are not always at hand – real datasets often are not very easy to acquire and sometimes, those existing ones lack those specific features. To solve this problem, some researchers resort to synthetic data generation tools but those software often require professional programming skills to hard code the required features into statistical properties. What's more, even with real datasets at hand, scientists sometimes want to edit them, clean them or modify them without touching for their own needs. For this purpose, the tools they use would ideally operate in the same interface where they can monitor the changes while editing, exploring and analysing their data.

Here we propose our scatterplot based data generation tool [95]. It uses the same visual interface for data generation and visualization such that users can monitor the whole progress without switching back and forth between different interfaces. This provides better context for later iterations of the data generation process and facilitates a more streamlined workflow. As users are able to create datasets more quickly, they can explore and generate a larger number of these and possibly more complex ones. This in turn will favor the development of more robust algorithms and software for high-dimensional data analysis and visualization. Similarly, as users are able to edit data and artifacts more informed and thoroughly they will be able to make faster progress in their data analysis efforts. We address this interface as 'WYSIWYG (What You See Is What You Get)' because it lets users directly manipulate the data in the same interface and also sculpt in 3D and draw in 2D.

Our system tool can operate either in a scatterplot matrix or with an interface for arbitrary projections [69]. As it is difficult for users to imagine what the effect of a specific carving operation done from one vantage point looks like from other vantage points, especially when the number of dimensions is larger than three, we provide simultaneous views from other vantage points of interest that reflect these shape changes.

6.2 SYSTEM OVERVIEW

Our interface is shown in Fig. 37. It consists of the following components:

Scatterplot display: This window shows the projected N-D data and serves as the main editing canvas. The data are projected as points into the 2D basis formed by the *projection plane axis (PPA) vectors*. These two orthogonal N-D vectors define the *x*- and *y*-axes of the resulting scatterplot display. The projected data axes can be optionally visualized directly in the scatterplot display or in isolation in the **data axes vector display**. The **vector component bar chart display** conveys more information about the dataset. The top two charts show the *x* and *y* components of the current PPA vectors while the bottom chart shows the PCA spectrum of the data.

N-D touchpad polygon: This is the interface by which the orientation of the PPA vector basis can be interactively configured, which in turn determines the projective view onto the data. Each polygon vertex represents a data dimension vector. The PPA vectors are determined by the positions of the red and blue points in the polygon's interior using generalized barycentric interpolation [65]. The **touchpad polygon control panel** lets the user switch between moving the red point (PPA x-axis) or the blue point (PPA y-axis).

Scatterplot display controls: This panel allows users to change the scatterplot's point size, zoom in or out, view the points in a 2D or 3D display, and see the moving trajectory as motion blur for better shape perception in motion parallax.

Sketching controls enable users to input the number of dimensions, the number of points, choose to start from scratch or with an existing dataset, and modify the points in axis-aligned or non-axis aligned fashion. **Brushing controls** let the user switch between brushing and erasing mode, choose brush size, brush density and brush color.

Scatterplot matrix (SPLOM) window: This is a separate window that is used when editing the distributions in axis-aligned mode. (see Section 5.3.1).

Scatterplot views window: This is another separate window that is used when editing the distributions in non-axis aligned mode. (see Section 5.3.2)

Distribution designer: Users design an initial distribution using this window (see Fig. 39 and explanatory text in Section 5.3.1, Step 1).

The design process begins with the user drawing an initial 2D distribution shape using the distribution designer. This 2D distribution is conceptually due to a collection of N-D points projecting into this shape. Initially these points would be randomly distributed in the other *N*-2 dimensions. Now we can pick another scatterplot projection and carve the 2D projected point cloud into one that we like to see from this orientation, under the constraint of the first drawing. We can repeat this for other projections and so on. This carving is essentially a subtraction of points from the current N-D distribution. However, a fundamental problem related to the nature of high-dimensional space is that the carving may subtract points in undesired locations in other projections. Therefore we need to replenish the N-D distribution every once in a while to satisfy all prior constraints, i.e. the shapes drawn and carved so far. Our proposed algorithm is presented

```
Point Generation
♦ Distribution painting in 2D
        Sketch shape, center line, and distribution profile
        System computes probability map
♦ Distribution backprojection
        System generates N-D points according to probability map
        (fully randomized in dimensions perpendicular to view plane)
♦Repeat as desired (allow multi-view painting)
Point Sculpting
♦ Distribution carving
        Manually erase unwanted points from other projection planes
        (this can delete points in other views in undesired locations)
        Update probability maps on these projection planes
♦ Distribution repair
        Manually replenish points erased in undesired locations
        (use probability maps constructed earlier to guide the process)
♦Repeat as desired
```

Fig. 38. Scatterplot based data generation algorithm.

in Fig. 38. We will demonstrate this algorithm first for the axis-aligned case and then for the non-axis aligned case. When interacting, users can switch among these modes at will.

6.3 METHODS

6.3.1 INTERACTIONS WITH AXIS-ALIGNED SCATTERPLOTS

Axis-aligned interactions make use of the SPLOM window for high-D space visualization. While this does allow for the generation (and editing) of distributions of any dimensionality, it requires them to be axis-aligned -- the non-axis aligned interface discussed in Section 5.3.2 removes this constraint. As outlined in Fig. 38, the point generation stage consists of two operations: distribution painting and backprojection. The sculpting supports two operations as well: distribution carving and repair. Editing operations use the same framework – just now a distribution already exists and does not have to be initialized. Editing an existing dataset is a good idea even when data generation is the goal – one does not have to start from scratch. We now describe each of these four operations in turn, using Fig. 40as a running example.

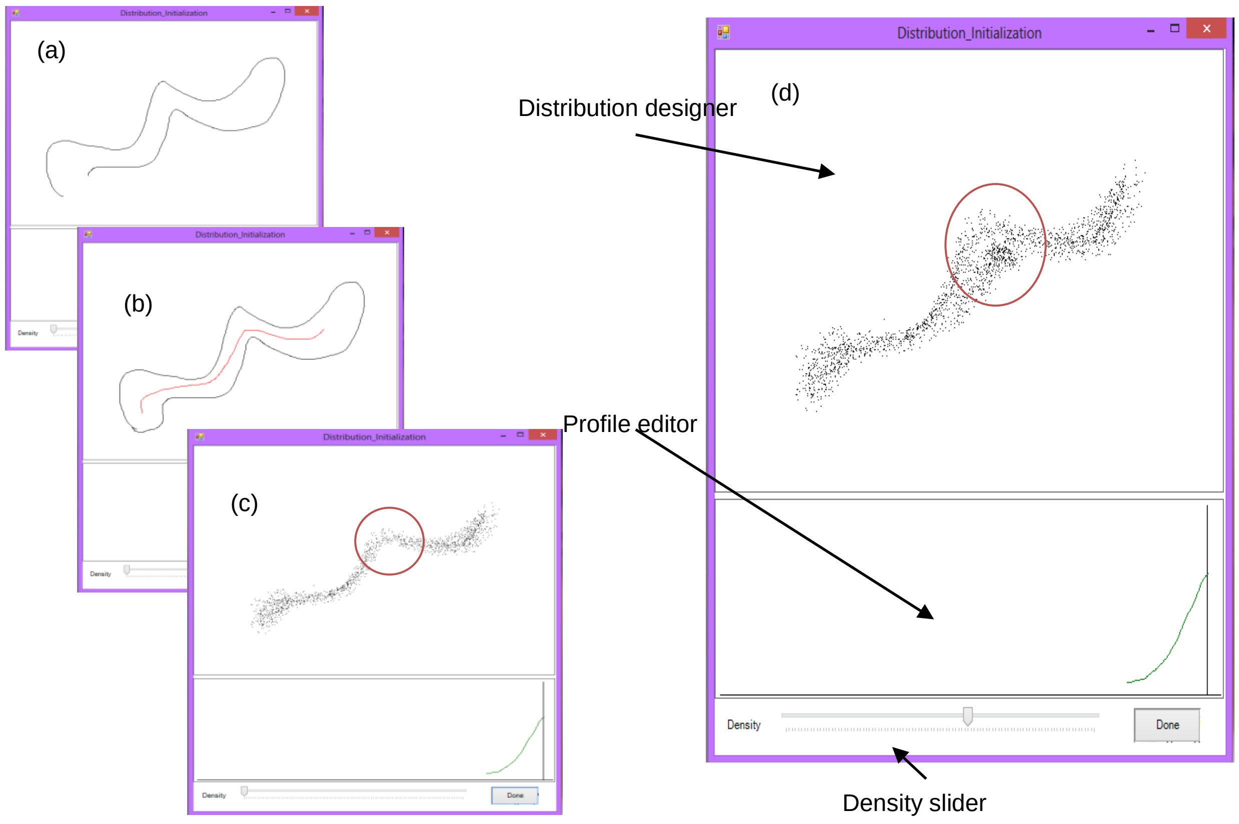
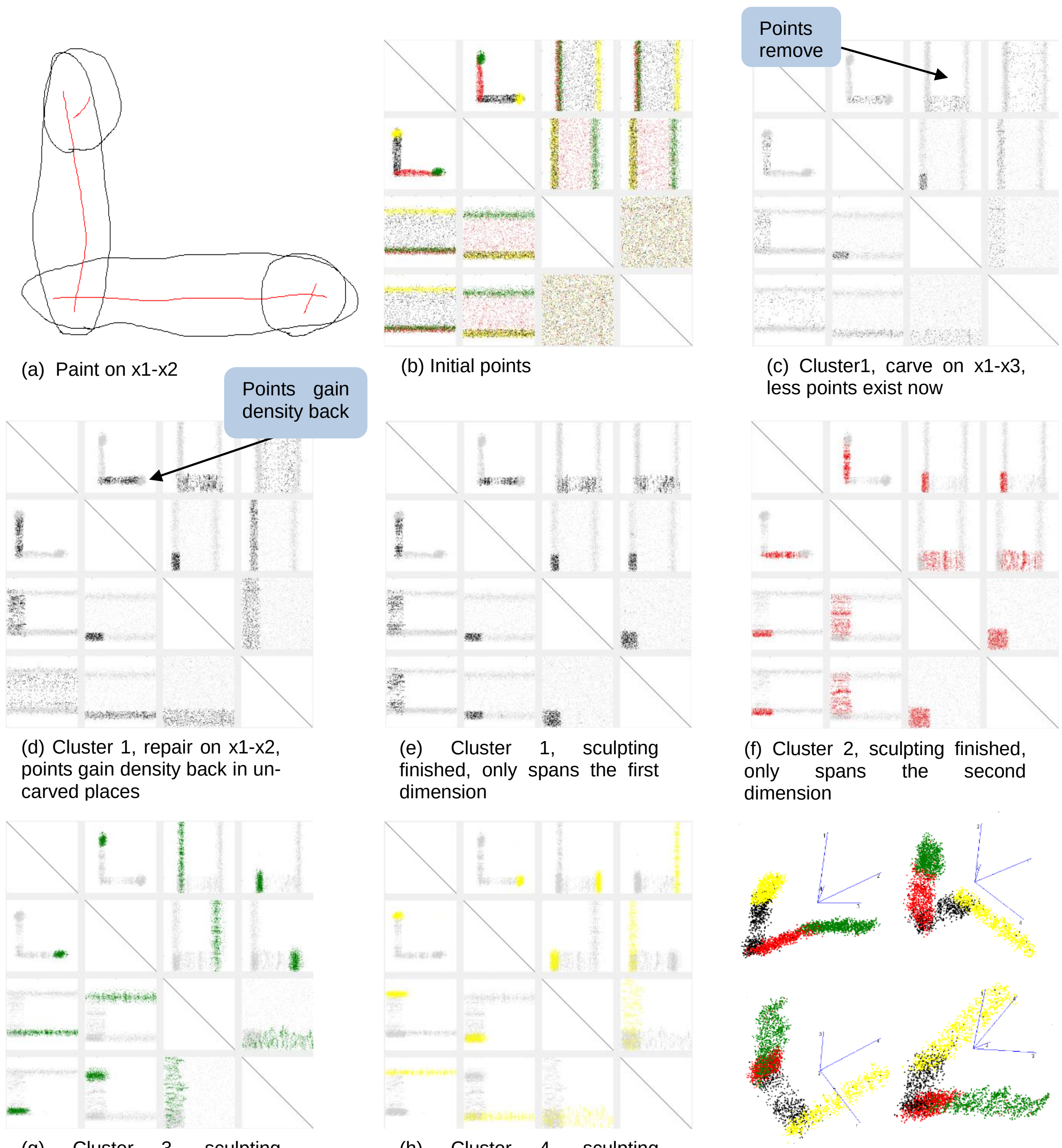


Fig. 39. Scatterplot sketching process. (a) Draw the boundary of the distribution. (b) Draw the center of the distribution. (c) Draw the profile of the distribution and the backprojected points. (d) Use the density bar to control the brush density and add more points.

Step 1: Distribution Painting. This activity occurs in the distribution designer window (see Fig. 39). The user first selects the desired axis-aligned view by placing the two PPA vector points in the N-D touchpad polygon onto the respective vertices. He then uses the sketching brush to draw the boundary of the distribution (Fig. 40(a)) and then the centerline (Fig. 40(b)). The center line is where the densest points should appear while the boundary constrains the range of the points. Next he uses the distribution profile designer to define the profile of the distribution (Fig. 40(c)). This fills the drawn shape with a point distribution which looks fairly regular. The user now has the option to paint additional points to make this distribution look more natural, i.e., less regular using a paint brush. The density of these drawn points can be controlled using a slider (Fig. 40(d)). Alternatively, points can also be removed using a fuzzy eraser, which strength is also configured by the density slider. The resulting 2D distribution is then converted into a probability map that governs how N-D points will project onto this projection plane. To ensure a sufficiently smooth probability map, we could optionally filter this density map with a 2D Gaussian function before backprojection. Figure 10a shows our running example in the SPLOM window – we sketched the shapes and centerlines of four clusters.



(a) Paint on x1-x2

(b) Initial points

(c) Cluster1, carve on x1-x3, less points exist now

(d) Cluster 1, repair on x1-x2, points gain density back in un-carved places

(e) Cluster 1, sculpting finished, only spans the first dimension

(f) Cluster 2, sculpting finished, only spans the second dimension

(g) Cluster 3, sculpting finished, only spans the third dimension

(h) Cluster 4, sculpting finished, only spans the fourth dimension

(i) result points, 4D snake shape

Fig. 40. An illustrative example of the scatterplot based data generation algorithm using axis-aligned views in the SPLOM window (panel (i) shows four views in the scatterplot window).

Step 2: Distribution Backprojection. Once the probability map has been constructed, the system uses it to generate *P* N-D points where *P* can be specified by the user. For the axis-aligned case, since all other projection planes at this initial configuration have a projection probability map with uniform distribution we can randomize the coordinates of all other *N*-2 dimensions. The result of this step is a set of *P* N-D data points with all coordinates defined. Fig. 40(b) shows the result of this step for a 4-D scatterplot matrix [32], composed of dimensions *x1*, *x2*, *x3*, and *x4*. Since the painting occurred in the *x1*-*x2* plane (which is now fully defined), the *x3*-*x4* projection has still a random point distribution. All other projections are partially defined since they either include dimension *x1* or *x2*.

Different distributions drawn in step 1 can be treated as different clusters and colored differently, as shown in the scatterplot matrix window of Fig. 40(b). The coloring clearly distinguishes between each drawn distribution and lets the user quickly see the points belonging to each (sub-) cluster. For distributions that may be too complicated to define at once or span more than one view, it helps to decompose them into simpler distributions and use the color coding as a guide to work on them one by one. For example, the final cluster shown in Fig. 40(i) (using four views in the scatterplot display) spans the entire 4D space but is composed of four different-colored sub-clusters each spanning just one dimension.

To create even more complex clusters the user may repeat the first two steps as often as desired by placing the PPA vector points on other vertices in the N-D touchpad polygon to select another view and paint on it. In this operation, all previously generated points are shown as inactive background labeled in gray color (Fig. 42(b)). This multi-view painting mechanism allows users to define highly multivariate clusters. We found that these two levels of decompositions can help greatly in comprehending the N-D sculpting task.

This point generation procedure just described provides us with a set of initial points to further work on – especially in those views that have at least one dimension not part of the painting process. Users can now sculpt on those dimensions to finish the data generation task.

Step 3: Distribution Carving. Fig. 40(c) visualizes the carving effects on all projections simultaneously in the scatterplot matrix window. In this figure the carving occurred for cluster 1 (black) in the *x1*-*x3* plane – all other three clusters are inactive now and are shown in gray. The user carves the points in the desired locations directly in the scatterplot display. The carving tool can also work as a fuzzy eraser such that every swipe only removes a random subset of the covered N-D points. Similar to the brush used in the data generation step, users can choose from three different eraser sizes. A large eraser allows quick point removal while a small eraser refines details. At every eraser location the algorithm picks a random set of points that project into a small box around it – the size of this box can be user specified – and these points will be removed from the location's current list of N-D points. Following, the updated distribution is stored in this projection's probability map. However, note that removing points from this projection's list will also remove these points from any other projections in the scatterplot matrix and update (scale down) their respective probability maps. This can lead to undesired effects and so the final (repair) step is required.

Step 4: Distribution Replenishing/Repair. As just mentioned, erasing/carving points in one view may delete points in other views in undesired places, and so we will have to bring these points back. Further, the repeated erasing of points will deplete the N-D distribution in general. Fig. 40(c) shows an example of this effect – the distribution of cluster 1 in the *x1-x2* plane is much weaker than in Fig. 40(b). We can repair/replenish the N-D distribution in two ways: (1) automatically by comparing the original projection probability maps with the current projections, and (2) user-driven by allowing the user to paint the missing points back on. The latter facility can also be used for general editing. In either mode, we need to make sure that we add points only in desired places. Given the current projection location subject to repair, we randomize the coordinates of the dimensions whose distributions have not been defined yet. For the other dimensions (excluding the two forming the current projection) – those for which the distributions have already been defined -- we compute their joint probability map and from it generate their coordinates. Fig. 40(d) shows an example where we observe that the distribution of cluster 1 has gained back the strength it had in Fig. 40(b).

Step 5 and on: Repeat. The user can pick any projection and update its probability map by carving or replenishing. This will activate the procedures described in detail in the previous steps. Fig. 40 (e)-(h) show a few more such operations for our demonstration dataset. The result is visualized in Fig. 40(i) using four non-axis aligned scatterplot projections. The generated data resembles a 4D snake-like structure which could not be constructed with existing tools that operate in 2.5D space.

6.3.2 Interactions with Non-axis Aligned Scatterplots

Non-axis aligned scatterplots provide additional freedom in cluster design but they require a more sophisticated navigation interface. A non-axis aligned view is selected by placing the PPA vector points inside the N-D touchpad polygon (see Fig. 41). We now describe the five data operations in turn.

Step 1: Distribution Painting: This is similar to the axis-aligned case, just now the selected view is non-axis aligned.

Step 2: Distribution Backprojection: This process is significantly more complicated than in the axis-aligned case. We first build a new coordinate system spanning the same data space using the Gram-Schmidt orthonormalization process [31]. The resulting vector set is the orthonormal set. In our case, we randomly generate (N-2) N-D vectors – we keep the two user selected vectors in order to preserve the shape the user painted. We also make sure that those vectors are linearly independent. Following we apply the Gram-Schmidt process on the N vectors. We treat the new orthonormal vectors as general unit base vectors and generate data using the same procedure as used in the axis-aligned case. Since the projection (painting) plane is not axis aligned, the generated point coordinates are described in the rotated space.

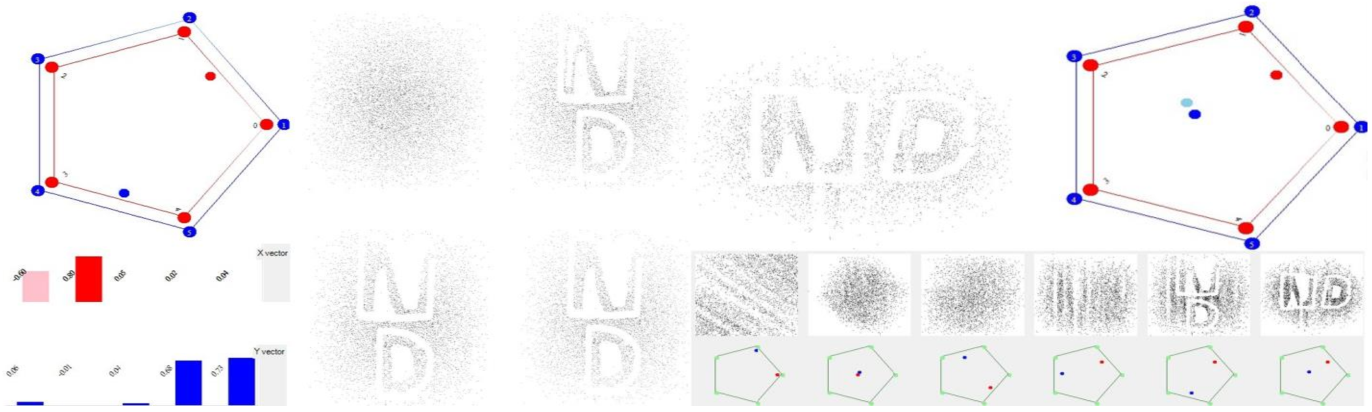

(a) (b) (c) (d)

Fig. 41. Non-axis aligned carving. (a) Top: the composition of the touchpad polygon. Bottom: the weighting of each dimension for the x and y PPAs. (b)(c) Carve and erase on one non-axis-aligned plane. (d) Select another plane to carve another shape. Note that the carved shapes can only be seen on those two selected views.

We can calculate their true data axis-aligned coordinates by multiplying them by the rotation matrix formed by the orthogonal basis vectors described above.

Step 3: Distribution Carving: As with the SPLOM in the axis-aligned case, we require a set of simultaneous views to enable users gain a more comprehensive understanding of the effect of the current carving. In the non-axis aligned case discussed here these views can be arbitrarily chosen by the user and typically hold views on certain structures the user would like to maintain. Then, any carving interaction in one view is immediately updated in all other views. Undesired effects can be undone by pressing an undo button and similar to an actual drawing process done on paper, we also provide a tool for erasing any unsatisfying mistakes.

Step 4: Distribution Replenishing/Repair: This is more problematic than for the axis-aligned case in which projections were mutually orthogonal. In the non-axis aligned case, unless two projections are proven to be orthogonal to each other, any view selected will partially overlap with another. Hence, when carving out points in one view, this effect cannot be repaired in another view using the above method. Our current solution is to allow the user to repair by bringing all points back, even if they have been carved out in the first view.

Step 5 and on: Repeat. This is similar to the axis-aligned case.

Fig. 41 demonstrates an example using an initial 5D Gaussian distribution. Our result dataset displays two characters ‘N’ and ‘D’ on the projection plane of which the x-axis is close to the first and second dimensions while the y-axis is close to the fourth and fifth dimensions. Fig. 41(a) shows the configuration of our polygonal touchpad. We move the red point close to the first and second dimensions and the blue point close to the fourth and fifth dimensions. The two bar charts below show the different dimension interpolation weights for the x and y-axis. We can clearly see that the first and second dimensions dominate the composition of the x-axis while the fourth and fifth dimensions have the largest value for y-axis. Fig. 41(b) shows the two multivariate scatterplots of the data on the previously selected projection plane, before and after carving. The top scatterplot shows a point cloud while the bottom displays the two characters after carving. This pattern can be observed only on this plane and it may not show any meaningful patterns on any other plane. Fig. 41(c) is a case in which the eraser comes in handy. The top scatterplot shows that a part in the 'N' is accidentally brushed away. Note the hollow portion on the left part of 'N'. Like in any painting tool, we can change to 'erasing' mode and use the eraser to bring those points back as shown in the bottom scatterplot. With the help of the brush and eraser users can create any pattern on any projection plane.

Next, the user may want to carve the existing set of characters such that they are also visible from another projection orientation. Carving on another projection plane may compromise the previous patterns so the user needs to be more careful. As is shown in Fig. 41(d), the user navigates to the second plane using the touchpad, adds this projection to the row of multiple views and starts carving. He can easily monitor the changes on the previous projection (the fifth) and make sure the ongoing carving does not adversely affect the overall shape of the data there.

We note that changing the carving plane can extend the intrinsic dimensionality of the resulting dataset to the number of dimension of the touchpad polygon. Our present example has a 5D polygon but there are no limits on how many vertices such a polygon can have. We note, however, that the ordering of the vertices is important with respect to the subspace of the data that can be reached [69]. We may initialize or reconfigure this vertex ordering using the parallel coordinate interface, but our navigation polygon also allows users to interchange, add, or remove vertices directly. Likewise, when a SPLOM is used and the dimensionality of the data grows high, the user may choose the dimensions composing the SPLOM using a selection interface.

6.3.3 Active and Inactive Points

The navigation polygon and the multiple views help users to easily carve data globally but not locally. Because of the high-dimensional space, a subset of data is cluttered with other data once the plane changes. Our interface also supports the concept of active and inactive points to provide the user complete control over selecting an active subset of data points for editing. Users can assign different point sets to clusters and control their status (active/non-active) via a check-box interface. In this case only the active points can be manipulated via the brushing/eraser tools. For example, all points marked in light gray in Fig. 40 are currently inactive and will be insensitive to any brush manipulation.

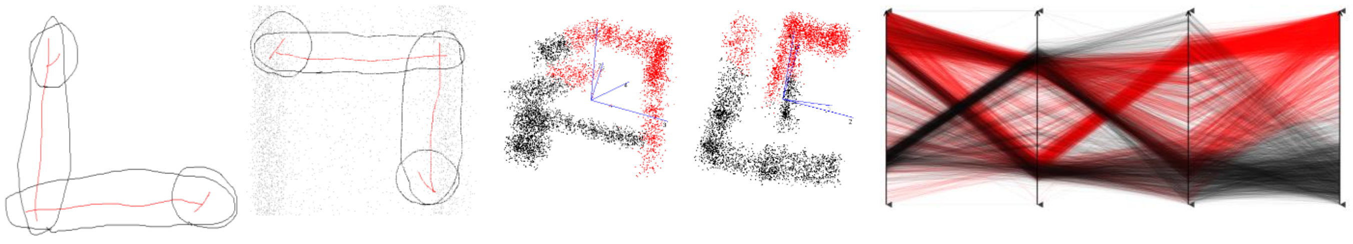

(a) (b) (c) (d) (e)

Fig. 42. Testing dataset generation. (a) Paint on *x1-x2* (b) Paint on *x3-x4* (c) (d) Generated data, clustered using standard k-means. Red and black are two clusters. (e) Generated data on parallel coordinates.

6.4 CASE STUDIES

6.4.1 Dataset Generation for Clustering Algorithm Testing

As mentioned, one of the most common problems when testing algorithms is the lack of a dataset that challenges a specific aspect of the algorithm. Let us take standard k-means which works well on linearly separable clusters, but how will it work on linearly non-separable clusters? To get insight we need a higher-D dataset that has these conditions. Now the challenge is where to get a high-D dataset with multiple clusters that are not linearly separable, even in 4D. We can quickly use our interface to generate such a 4D dataset.

We aim to create a dataset with two such clusters spanning the entire 4D space – one of these has an 'L' shaped structure on the *x1-x2* plane while the other has the same structure on the *x2-x3* plane. This guarantees the linear non-separability we strive for. Similar to Fig. 40, we decompose the two 4D clusters into four sub-clusters for ease of manipulation. The generation process of the first cluster is exactly the same as in Fig. 40. For the second cluster, we initially paint on the *x2-x3* plane, and also go through the sculpting procedure to make sure each sub-cluster only span one dimension. Fig. 42(a)(b) show the initially painted distributions, while Fig. 42(c)(d) show the two clusters as scatterplots. Fig. 42(b) shows those points generated from the first distribution painting as background gray points.

We then performed standard k-means clustering on the generated dataset, and we obtained two clusters colored red and black (Fig. 19(c)(d)). From the parallel coordinate's view of this dataset (Fig. 42(e)) we observe that the standard k-means clustering mainly operated on the 4th dimension and that the resulting two clusters are not accurate. We now know that a more sophisticated clustering algorithm, such as a kernel method, must be used.

6.4.2 Existing Data Editing

Our proposed system also allows users to modify an existing data set for better algorithm testing and data analysis purposes.

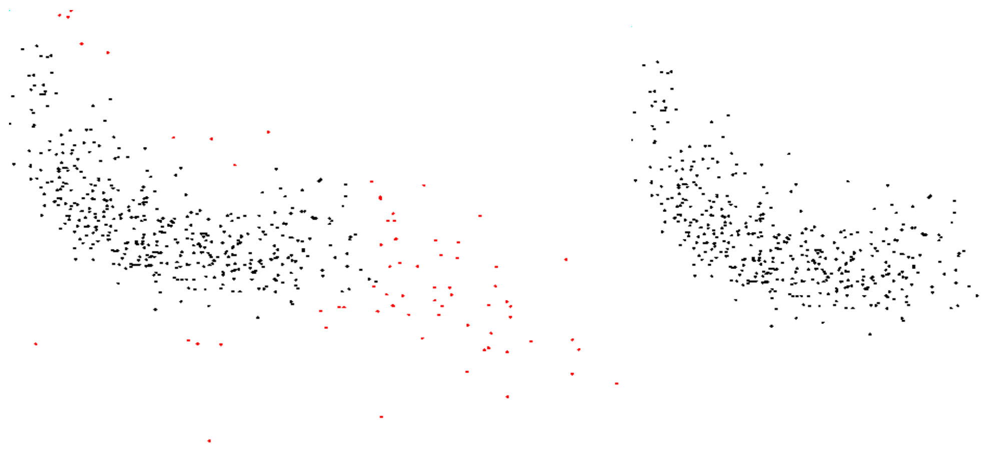


Fig. 43. Edit an existing data set. (a) Red points are outliers (b) Outliers removed.

One of the most encountered problems in data analysis is the existence of outliers. Outliers may impair the ability to discover underling patterns or finding meaningful trends. One may use outlier detection algorithms to remove them, but different algorithms are based on different definitions (density based or distance based) of outliers and may not fit into a particular need. Hence, to be able to visually detect outliers and subsequently remove these outliers would be more suitable in the general case.

Let us take the housing data set [119] as an instance. A user might want to determine the relationship between the 'average number of rooms per dwelling' and the '% lower status of the population'. After plotting the data against these two attributes (Fig. 42(a)), we see that most of the points are concentrated on the main cluster but outliers (Fig. 43(a), red points) exist. This makes further analysis hard, and simply including all the points in the analysis would introduce errors. Some methods might be able to adjust to the shape of the data, but they often bring the risk of over fitting. Pre-processing the data to remove unwanted points would be very useful. The user could employ general outlier detection algorithms but since this is a high-D data set, the importance of these two attributes might be reduced. In addition, these algorithms often require a pre-setting of some parameters or thresholds which can lead to erroneous results. Our system would come in handy here. The user could navigate to the desired view and visually detect outliers according to their own understanding and need, brushing away unwanted points (Fig. 43(b)). Using the remaining points for further analysis would then yield more robust results.

CHAPTER 7

CONCLUSION

In this dissertation we have introduced innovative methods to present, analyze and generate high dimensional data. We started with an intrinsic dimensionality study. We first used visually aided DBSCAN to cluster the data and then perform PCA on each cluster to extract the subspaces they reside. The results shows that most clusters reside in subspaces that are composed of more than two principal components. This justifies our research direction that incorporating a third dimension in to high dimensional non spatial data visualization is indeed helpful.

We then represented an advanced interactive data exploration tool – the Subspace Voyager. It decomposes the high dimensional space into a series of salient 3D subspaces and provides the users with both automatic and manual means to analyze those subspaces. Our Subspace Voyager essentially makes use of motion parallax and gives the users the illusion of a third dimension that helps to disambiguate the data.

For the automatic methods, the Subspace Voyager uses Ant Colony Optimization algorithm to offer the users with meaningful views that is close to their starting projections according to their own needs. For the manual interactions, it lets users operate directly on a familiar trackball interface. To examine the data locally, users can simply rotate the trackball and if they want to go to adjacent subspaces, the ‘chase cluster’ functionality allows them to rapidly change the weightings of different data dimensions.

The Subspace Voyager also uses a trail map to save explored views. The saved views are automatically spread out and they sizes can be changed by a slider. Any anytime, users can bring any previously saved view back to the trackball for further exploration. To let the users smoothly present their findings to others, a path can be designed between key views and system will display those key findings one after another, in a continuous fashion.

We have also proposed a novel 3D shaded shape representation (SSR) for non-spatial data. Our representation uses the Marching Cubes algorithm to build isosurfaces enclosing the data points according to a user-defined isovalue. Users can change the transparency, the rendering mode, and the level of the detail of the generated shapes, as well as the lighting mode of the scene. Case studies we presented showes that our SSR can be advantageous over traditional scatterplot displays. It allows users to better discern relations in the dataset, and it also allows users reveal structures that scatterplots cannot discover. Finally, we also find, through a study conducted in Amazon Mechanical Turk, that mainstream users prefer our 3D representation over scatterplots. This has important implication for how general users can be engaged into reasoning with data. Our results suggest that it might be better to provide 3D shaded surface displays for this purpose.

In the future, we seek to make the Marching Cubes algorithm parallel. The MC algorithm is currently implemented in JavaScript and this stands in the way of achieving interactive performance when shifting shapes to adjacent subspaces. This could be achieved via GPU acceleration. Further, we would also like to solve issues with color blending. When a shape is shown behind another semi-transparent shape, false color can be generated in the overlap area. We are considering using the method described in [52] to overcome this problem. Finally, when the shapes are displayed in semi-transparent multi-layer mode, as far as the data sets we encountered so far, the outermost layer has always been the most transparent one because the data has dense points in the middle and sparse ones along the boundary. But what if a dataset has the opposite property and has dense points along the border? How can we still reveal the inner structure in that case? We would like to look further into possible solutions for this case.

In the end, we presented our sketching based high dimensional data generation tool which is easy and intuitive to use even for novice users. The sketching interface is based on scatterplot and which is also used as the visualization interface. This design saves users from switching back and forth when generating data. Unlike previous data generation paradigms, our tool can truly generate high dimensional data. It also lets users to modify existing data. However, it does have its weakness too. One notable thing is the navigation pad is a bit more abstract. In our future work, we could substitute this interface with our trackball navigation system. We also want to add curve drawing and line drawing widgets for better user experience.

BIBLIOGRAPHY

[1] E. Abbot, *Flatland: A Romance of Many Dimensions*, Dover Thrift Edition, 1984.

[2] J. Ahn, J.S. Marron, K.M. Muller, Y. Chi, “The high-dimension, low-sample-size geometric representation holds under mild conditions”, *Biometrika* 94. (2007) 760–766.

[3] G. Albuquerque, M. Eisemann, T. Löwe and M. Magnor, “Hierarchical Brushing of High-Dimensional Data Sets Using Quality Metrics,” *Vision, Modeling, and Visualization,* 2014.

[4] G. Albuquerque, T. Löwe, and M. Magnor, "Synthetic generation of high-dimensional datasets," *IEEE Trans on Visualization and Computer Graphics,* 17(12), pp. 2317-24, 2011

[5] E. Amorim, E. Brazil, J. Daniels, P. Joia, L. Nonato, M. Sousa, "iLAMP: Exploring high-dimensional spacing through backward multidimensional projection," in *IEEE VAST,*, 2012.

[6] A. Anand, L. Wilkinson, T. Dang. "Visual pattern discovery using random projections." *Visual Analytics Science and Technology (VAST),* pp. 43-52, 2012.

[7] E. Angel, D. Shreiner, Interactive Computer Graphic with WebGL (7th Edition), Addisoin-Wesley, 2014.

[8] D. Asimov, "The Grand Tour: A Tool for Viewing Multidimen-sional Data," *SIAM J. Scientific and Statistical Computing,* vol. 6, no. 1, pp. 128-143, 1985.

[9] I. Assent, R. Krieger, E. Müller, T. Seidl, “VISA: visual subspace clustering analysis,” *ACM SIGKDD Explorations Newsletter*, 9(2), 5-12, 2007.

[10] T. Baudel, "From information visualization to direct manipulation: extending a generic visualization framework for the interactive editing of large datasets,"*Proc of the ACM symposium on User interface software and technology,* 2006

[11] R. Bellman, “Adaptive Control Processes: A Guide Tour,” Princeton University Press, Princeton, NJ, 1961.

[12] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: applications to image and text data," *Proc. of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, 2001.

[13] S. Card, J. Mackinlay, B. Shneiderman. Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann, Jan 25, 1999.

[14] G. Z. Chen, J. Q. Wang and C. J. Li, "Solving the Optimization of Projection Pursuit Model Using Improved Ant Colony Algorithm," in *Fourth International Conference on Natural Computation*, 2008.

[15] J. Choo, S. Bohn, H. Park, "Two-stage framework for visualiza-tion of clustered high dimensional data," *Proc. IEEE Visual Analytics Science and Technology Conference (VAST),* pp. 67 - 74, 2009.

[16] D. Cook, A. Buja, J. Cabrera & C. Hurley, "Grand Tour and Projection Pursuit," *Journal of Computational and Graphical Statistics,* vol. 4, no. 3, pp. 155-172, 1995.

[17] D. Cook, E.K. Lee, A. Buja, H. Wickham, "Grand Tours, Projection Pursuit Guided Tours and Manual Controls," in *Handbook of Data Visualization*, New York, Springer, 2006.

[18] R. DeMillo and A. Offutt, "*Constraint-Based Automatic Test Data Generation*," *IEEE Trans on Software Engineering*, vol. 17, no. 9, pp. 900-9101, 1997.

[19] M. Dorigo, "Optimization, Learning and Natural Algorithms," in *PhD Thesis*, Politecnico di Milano, 1992.

[20] W.P. Elderton, “"Tables for Testing the Goodness of Fit of Theory to Observation,” *Biometrika.*, 1 (2): 155–163, 1902.

[21] N. Elmqvist, P. Dragicevic, J.-D. Fekete, "Rolling the dice: multi-dimensional visual exploration using scatterplot matrix navigation," *IEEE Trans. Visualization and Computer Graphics,* vol. 14, no. 6, pp. 1539-1148, 2008.

[22] M. Ester, H. Kriegel, J. Sander, X. Xu. "A density-based algorithm for discovering clusters in large spatial databases with noise." *KDD*, vol. 96, pp. 226-231. 1996.

[23] R. Ferguson and B. Korel, “The chaining approach for software test data generation,” ACM Transactions on Software Engineering and Methodology (TOSEM), 5(1*), pp63-86*, 1996

[24] M. D. Flood, V. L. Lemieux, M, Varga and B.L. W. Wong, “The Application of Visual Analytics to Financial Stability Monitoring,” *Records and Information Management for Financial Analysis and Risk Management Workshop,* 2011

[25] S. L. France, J. D. Carroll and H. Xiong. “Distance Metrics for High-Dimensional Nearest Neighborhood Recovery: Compression and Normalization,” *Information Sciences* 184 (2012) 92–110.

[26] M. Friendly, "A Brief History of Data Visualization," in *Handbook of Data Visualization*, Springer Berlin Heidelberg, pp. 15-56, 2008.

[27] J. Friedman, J. Tukey, "A projection pursuit algorithm for ex-ploratory data analysis," *IEEE Trans. on Computers,* 23(9): 881-890, 1974.

[28] A. Gilbert, L. Piras, J. Wang, F.Yan, E. Dellandrea, R. Gaizauskas, M. Villegas and K. Mikolajczyk, “Overview of the ImageCLEF 2015 Scalable Image Annotation, Localization and Sentence Generation task.” *CLEF2015 Working Notes. CEUR Workshop Proceedings*, CEUR-WS.org, Toulouse, France, 2015.

[29] A. Gotlieb, B. Botella, and M. Rueher. "Automatic test data generation using constraint solving techniques." *ACM SIGSOFT Software Engineering Notes*. Vol. 23. No. 2. ACM, 1998.

[30] S. Gratzl, A. Lex, N. Gehlenborg, H. Pfister and M. Streit, "Lineup: Visual analysis of multi-attribute rankings", *IEEE TVCG*, vol. 19, no. 12, pp. 2277-2286, 2013.

[31] W. Greub, Linear Algebra (4th ed.), Springer, 1975.

[32] J. Hartigan, "Printer graphics for clustering," *Journal of Statistical Computation and Simulation*, 4(3):187-213, 1975.

[33] G. Hatfield, "Perception as Unconscious Inference," in *Perception and the Physical World: Psychological and Philosophical Is-sues in Perception*, D. Heyer and R. Mausfeld, Wiley, 2002, pp. 115-143.

[34] H. Helmholtz, “Handbuch der Physiologischen Optik (Dritte Auflage, Dritter Band),” Verlag von Leopold Voss, Hamburg/ Leipzig, Germany, 1911.

[35] P. Hoffman, "*DNA visual and analytic data mining*," *Proceedings of the IEEE Visualization*, pp. 437-441, 1997.

[36] P. Hoffman and G. Grinstein, “Visualizations for High Dimensional Data Mining - Table Visualizations.

[37] S. Ingram, T. Munzner, M. Olano, "*Glimmer: Multilevel MDS on the GPU*," *IEEE Trans. Visualization and Computer Graphics*, vol. 15, no. 2, pp. 249-261, 2009.

[38] A. Inselberg, "The Plane with Parallel Coordinates". *Visual Computer* 1 (4): 69–91, 1985.

[39] S. Johansson, J. Johansson, "*Interactive dimensionality reduction through user-defined combinations of quality metrics*," *IEEE Trans. Visualization and Computer Graphics,* vol. 15, no. 6, pp. 993-1000, 2009.

[40] I.T. Jolliffe. 1986. Principal Component Analysis. Springer-Verlag, Berlin.

[41] E. Kandogan, "Star Coordinates: a multi-dimensional visualization technique with uniform treatment of dimensions," *IEEE InfoVis Late Breaking Topics*, 2000.

[42] Y.A. Kang, C. Görg, and J. Stasko. "How can visual analytics assist investigative analysis? Design implications from an evaluation," *Visualization and Computer Graphics, IEEE Transactions on* 17, no. 5 (2011): 570-583.

[43] D. Keim, G. Andrienko, J. Fekete, C. Goerg, J. Kohlhammer, and G. Melancon, “Visual Analytics: Definition, Process, and Challenges,” Springer Berlin Heidelberg, 2008.

[44] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler, “Visual Analytics: Scope and Challenges,” *Visual Data Mining: Theory, Techniques and Tools for Visual Analytics.* Springer, 2008.

[45] D. A. Keim, M. C. Hao, U. Dayal and M. Hsu” Pixel bar charts: A visualization technique for very large multi-attribute data sets,” *Information Visualization*, 1 (1): 20 & 34, 2002.

[46] D. Keim, J. Kohlhammer, G. Ellis and F. Mansmann, “Mastering The Information Age – Solving Problems with Visual Analytics,” Eurographics Association.

[47] H. Kim, J. Choo, H. Park and A. Endert, “InterAxis: Steering Scatterplot Axes via Observation-Level Interaction,” *IEEE Trans. Visualization and Computer Graphics,* 22(1):131-140, 2015.

[48] C. Kinney, "Heat Map". United States Patent and Trademark Office Patent 75263259, 01 09 1993.

[49] H. Kriegel, P. Kröger, A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *ACM Trans. on Knowledge Discovery from Data*, 3(1):1, 2009.

[50] J. Kruskal, M. Wish, "Multidimensional Scaling," Sage Publications, 1977.

[51] T. Kohonen, "*The Self-Organizing Map,*" *Proceeding of the IEEE,* vol. 78, no. 9, 1990.

[52] L. Kühne, J. Giesen, Z. Zhang, S. Ha and K. Mueller, "Data-Driven Approach to Hue-Preserving Color-Blending," *IEEE Transactions on Visualization and Computer Graphics*, 18(12): 2122-2129, 2012.

[53] P. M. Kroonenberg, "Introduction to Biplots for G×E Tables."

[54] M. Kurimo, "Indexing audio documents by using latent semantic analysis and SOM," in *Kohonen Maps, pp 363-374*, Elsevier, 1999.

[55] E. Lee, D. Cook, S. Klinke, T. Lumley, "Projection pursuit for exploratory supervised classification," *J Computational and Graphical Statistics,* 14(4):831-846, 2005.

[56] J. Lee, K. T. McDonnell, A. Zelenyuk, D. Imre and K. Muller, "A Structure-Based Distance Metric for High-Dimensional Space Exploration with Multi-Dimensional Scaling", *IEEE Trans on Visualization and Computer Graphics,* 20(3): 351-364, 2014.

[57] J. Li, J. Martens, J. van Wijk, "Judging correlation from scatter-plots and parallel coordinate plots," *Information Visualization,* vol. 9, no. 1, pp. 13-30, 2010.

[58] L. Lipton, *Foundations of the Stereoscopic Cinema - A Study in Depth* (p. 56). New York: Van Nostrand Reinhold, 1982.

[59] S. Liu, B. Wang, J.J. Thiagarajan, P.-T. Bremer, V. Pascucci. "Visual Exploration of High-Dimensional Data through Subspace Analysis and Dynamic Projections." *Comput. Graph*. Forum. 34(3): 271-280, 2015.

[60] W. E. Lorensen and H. E. Cline, "Marching Cubes: A high resolution 3D surface construction algorithm," *Computer Graphics,* 21(4), 163-169, 1987.

[61] L. van der Maaten, G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research,* vol. 9, no. 11, pp. 2579-2605, 2008.

[62] P. McMinn, "Search-based Software Test Data Generation: A Survey," *Software Testing, Verification and Reliability,* 14(2), pp. 105-156, 2004.

[63] A. Mayorga and M. Gleicher, "Splatterplots: Overcoming overdraw in scatter plots", *IEEE Transactions on Visualization and Computer Graphics,* vol. 19, no. 9, pp. 1526-1538, 2013.

[64] G. McLachlan, "Discriminant Analysis and Statistical Pattern Recognition," *Wiley Interscience*, 2004.

[65] M. Meyer, H. Lee, A. Barr, M. Desbrun, "Generalized barycentric coordinateson irregular polygons", *Graphics Tools*, 7(1):13-22, 2002.

[66] C. Michael, G. McGraw, M. Schatz, R. Walton, R. Res, and V. Sterling, "Genetic Algorithms for Dynamic Test Data Generation," *Automated Software Engineering,* pp. 307-308, 1997.

[67] W. Miller and D. L. Spooner. "Automatic generation of floating-point test data." *IEEE Transactions on Software Engineering* 2.3 (1976): 223.

[68] E. Nam, Y. Han, K. Mueller, A. Zelenyuk, D. Imre, "ClusterSculptor: A Visual Analytics tool for high-dimensional data," *IEEE VAST,* pp. 75-82, 2007.

[69] E. Nam and K. Mueller, "*TripAdvisorND: A Tourism-Inspired High-Dimensional Space Exploration Framework with Overview and Detail*," *IEEE Trans. on Visualization and Computer Graphics,* vol. 19, no. 2, pp. 291-305, 2013.

[70] C.H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala, "Latent semantic indexing: A probabilistic analysis," *Proc. 17th ACM Symp. on the Principles of Database Systems, pp 159 - 168*, 1998.

[71] E. Papenhausen, B. Wang, M. H. Langston, B. Meister, M. Baskaran, T. Henretty, T. Izubuchi, A. Johnson, C. Jung, M. Lin, K. Mueller and R. Lethin, "Polyhedral User Mapping and Assistant Visualizer Tool for the R-Stream Auto-Parallelizing Compiler," *the 3rd IEEE Working Conference on Software Visualization.*

[72] R. Pargas, M. Harrold, and R. Peck, "Test-Data Generation Using Genetic Algorithms," *Software Testing, Verification and Reliability,* pp. 41-48, 1999.

[73] R. S. Pilling, "An Introduction to SAS® Visual Analytics for the Health Care Sector."

[74] C. Posse, "An Effective Two-dimensional Projection Pursuit Algorithm," in *Communications in Statistics: Simulation and Computation 19*, pp. 1142-1164.

[75] G.G. Robertson, J.D. Mackinlay and S.K. Card, "Cone Trees: Animated 3D Visualizations of Hierarchical Information," *Human Factors in Computing Systems, CHI* '91 Conf. Proc., pp. 189-194, 1991.

[76] P. Ruchikachorn, K. Mueller, "Learning Visualizations by Analogy: Promoting Visual Literacy through Visualization Morphing," *IEEE Trans. Visualization and Computer Graphics*, 21(9):1028-1044, 2015.

[77] M. Schäfer, L. Zhang, T. Schreck, A. Tatu, J. Lee, M. Verleysen, D. Keim, "Improving projection-based data analysis by feature space transformations," *SPIE Electronic Imaging,* 2013, pp. 86540H-86540H.

[78] C. Schmid, H. Hinterberger, "Comparative multivariate visualiza-tion across conceptually different graphic displays," Proc. SSDBM, pp. 42 - 51, 1994.

[79] J. Seo, B. Shneiderman, "A rank-by-feature framework for unsu-pervised multidimensional data exploration using low dimensional projections," *Proc. IEEE Info Vis,* pp. 65-72, 2004.

[80] L. Shao, D. Sacha, B. Neldner, M. Stein and T. Schreck, "Visual-Interactive Search for Soccer Trajectories to Identify Interesting Game Situations," *Electronic Imaging, Visualization and Data Analysis,* pp. 1-10(10), 2016

[81] P. Shirley, S. Marschner, *Fundamentals of Computer Graphics*, A K Peters/CRC Press, 2015.

[82] M. Sips, B. Neubert, J. Lewis, P. Hanrahan, "Selecting good views of high-dimensional data using class consistency," *Computer Graphics Forum,* vol. 28, no. 3, pp. 831-838, 2009.

[83] K. Socha, M. Dorigo, "Ant colony optimization for continuous domains," *European Journal of Operational Research 185,* pp. 1155-1173, 2008.

[84] D. Stodder, “Visual Analytics for Making Smarter Decisions Faster. Applying Self-Service Business Intelligence Technologies to Data-Driven Objectives,” *TDWI Best Practice Report*.

[85] D. Swayne, D. Lang, A. Buja and D. Cook, "GGobi: evolving from XGobi into an extensible framework for interactive data visualiza-tion," *Comp. Statistics & Data Analysis*, 43(4):423-444, 2003.

[86] A. Tatu, G. Albuquerque, M. Eisemann, P. Bak, H. Theisel, M. Magnor, D. Keim, "Automated analytical methods to support visual exploration of high-dimensional data," *IEEE Trans. on Visual-ization and Computer Graphics*, 17(5): 584-597, 2011.

[87] A. Tatu,, L. Zhang, E. Bertini., T. Schreck,, D. Keim, S. Bremm, T. von Landesberger, “Clustnails: Visual analysis of subspace clusters,” *Tsinghua Science and Technology,* 17(4), 419-428, 2012.

[88] J. Tenenbaum, V. De Silva, J. Langford. "A global geometric framework for nonlinear dimensionality reduction." *Science,* (290) 5500: 2319-2323, 2000.

[89] N. Tracey, J. Clark, K. Mander, and J. McDermid.”An automated framework for structural test-data generation,” *13th IEEE International Conference on Automated Software Engineering, pp 285-288, 1998*.

[90] M. Vartak, S. Rahman, S. Madden, A. Parameswaran, N. Polyzotis, “SeeDB: efficient data-driven visualization recommendations to support visual analytics,” *Proceedings of the VLDB Endowment*. 8(13*), pp 2182-93,* 2015.

[91] S. Verheyen, E. Ameel, G. Storms. "Determining the dimensionality in spatial representations of semantic concepts," *Behavior Research Methods*, 39(3): 427-438, 2007.

[92] B. Wang. K. Mueller, "Does 3D really make sense for visual cluster analysis? Yes!" *International Workshop on 3DVis: Does 3D Really Make Sense for Data Visualization?* Paris, France, November 2014.

[93] B. Wang, K. Mueller, “Shape Up: Are Shaded Surface Displays More Appealing and Informative than Scatterplots?” in submission.

[94] B. Wang, K. Mueller, “The Subspace Voyager: Exploring High-Dimensional Data along a continuum of Salient 3D Subspaces,” *IEEE Trans. on Visualization and Computer Graphics,* under revision.

[95] B.Wang, P. Ruchikachorn, K. Mueller, "SketchPadN-D: WYDIWYG Sculpting and Editing in High-Dimensional Space," *IEEE Transactions on Visualization and Computer Graphics,* 19(12):2060-2069, 2013.

[96] C. Ware, Information Visualization: Perception for Design, Morgan Kaufman, 2000.

[97] J.J. van Wijk and H. van de Wetering, “Cushion Treemaps: Visualization of Hierarchical Information,” *Proc. IEEE Symp. Information Visualization,* pp. 73-78, 1999.

[98] P. Wong, R. Bergeron, "Multivariate visualization using metric scaling," *Proc. IEEE Visualization,* pp. 111-118, 1997.

[99] F. Yates, “Contingency table involving small numbers and the χ2 test,” *Supplement to the Journal of the Royal Statistical Society* 1(2): 217–235, 1934.

[100] X. Yuan, P. Guo, H. Xiao, H. Zhou, H. Qu, "Scattering points in parallel coordinates," *IEEE Trans. Visualization and Computer Graphics,* vol. 15, no. 6, pp. 1001-1008, 2009.

[101] X. Yuan, D. Ren, Z. Wang, and C. Guo, “Dimension Projection Matrix/Tree: Interactive Subspace Visual Exploration and Analysis of High Dimensional Data”, *IEEE Trans. on Visualization and Computer Graphics*, 19(12): 2625 – 2633, 2013.

[102] Z. Zhang, K. McDonnell, K. Mueller, "*A network-based interface for the exploration of high-dimensional data spaces*," in *IEEE Pacif-ic Vis Symposium,* Songdo, Korea, March, 2012.

[103] Z. Zhang, X, Tong, K. McDonnell, A. Zelenyuk, D. Imre, K. Mueller. "An interactive visual analytics framework for multi-field data in a geo-spatial context." *Tsinghua Science and Technology* 18(2): 111-124, 2013.

[104] Z. Zhang, B. Wang, F. Ahmed, IV Ramakrishnan, A. Viccellio, R. Zhao, K. Mueller, “The Five W’s for Information Visualization with Application to Healthcare Informatics”, *IEEE Transactions on Visualization and Computer Graphics,* 19 (11): 1895-1910, 2013.

[105] http://miegakure.com/ [Accessed 06/03/2016]

[106] https://onlinecourses.science.psu.edu/stat505/node/54 [Accessed 06/03/2016]

[107] College Prowler， http://collegeprowler.com, 2012. [Accessed 09/2009]

[108] US News Best Colleges ， http://colleges.usnews. rankingsandreviews.com, 2012. [Accessed 09/2009]

[109] http://www.imageclef.org/ [Accessed 06/03/2016]

[110] http://risenet.prhlt.upv.es/webupv-datasets/dwld/v2013/feats_visual/ [Accessed 06/03/2016]

[111] http://risenet.prhlt.upv.es/webupv/datasets/agreement?dwld=dwld/v2013/feats_visual/webupv13_devel_visual_getlf.feat.gz [Accessed 06/03/2016]

[112] http://www.cadaster.eu/node/67.html [Accessed 01/10/2015]

[113] MOE (The Molecular Operating Environment) Version 2008.10, software available from Chemical Computing Group Inc., http://www.chemcomp.com [Accessed 01/10/2015]

[114] https://archive.ics.uci.edu/ml/datasets/Iris [Accessed 03/16/2016]

[115] http://www.smithsonianmag.com/science-nature/do-our-brains-find- certain-shapes-more-attractive-than-others-180947692/?no-ist. [Accessed 03/16/2016]

[116] https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits.[Accessed 03/09/2016]

[117] https://archive.ics.uci.edu/ml/datasets/Image+Segmentation. [Accessed03/25/2016]

[118] http://colorvisiontesting.com/ishihara.htm. [Accessed 03/25/2016]

[119] http://archive.ics.uci.edu/ml/datasets/Housing. [Accessed 03/25/2016]

[120] https://www1.udel.edu/johnmack/frec682/cholera/ [Accessed 06/03/2016]

[121] https://en.wikipedia.org/wiki/Scatter_plot [Accessed 06/03/2016]

[122] https://github.com/OpenRefine [Accessed 06/03/2016]
[123] https://archive.ics.uci.edu/ml/datasets/Housing [Accessed 06/03/2016]