Interaction Design in Multidimensional Visualization

Techniques for multidimensional data visualization, exploration and visual analytics

Tze-Haw Huang

A thesis submitted for the degree of
Doctor of Philosophy in Computing Sciences
at the
University of Technology, Sydney

Faculty of Information Technology
University of Technology, Sydney
Sydney, Australia

2015

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

Date	**June, 2015**
Author	**Tze-Haw Huang**
Title	**Interaction Design in Multidimensional Visualization**
Degree	**Doctor of Philosophy in Computing Sciences**

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Candidate:

Date:

Acknowledgement

During the pursuit of higher education, I have taken on a number of roles, these being a part time Ph.D. candidate, a full time software engineer, a father and a husband. Without the support of many people it would have been very difficult for me to complete this dissertation.

First of all, a special thanks to my supervisor Associate Professor Dr. Mao Lin Huang, who contributed invaluable suggestions and knowledge to my work. I would never have been able to finish my Ph.D. without his exemplary and much appreciated guidance.

Secondly, I want to give my utmost thanks to my wife Yi Lau for her understanding, patience, care and continuous support throughout my PhD candidature. My sincere thanks is also extended to my parents for their consistent and very welcome encouragement over the duration of my doctoral studies.

Finally, I want to express my thanks to all of the people that I encountered who provided direct and indirect support while I worked on my dissertation.

Tze-Haw Huang
Sydney, September, 2014

Table of Contents

List of Figures

List of Algorithms

Abstract

Interaction is an overloaded term in information visualization. Basically, every software tool is interactive but mostly through the manipulation of a widget. Broadly speaking, a visualization is just a software application. What makes the interactive component of a visualization really distinctive is how well it supports an arbitrary selection of data directly in the interface in order to facilitate subsequent analytic tasks. This is challenging due to over-plotting and visual clutter in the multidimensional space and such phenomenon is commonly known as the *curse of dimensionality*.

Data selection is a frontier of a visualization and too many multidimensional visualizations claiming to be interactive mostly address the change of view without explicitly specifying the core technique of how to materialize such operations. Perhaps, the interactive component is achieved through the traditional widget.

To overcome the complexity of truly interacting with multidimensional data for effective visual analytics, we first propose an interactive framework for better understanding of the problem domains. Dynamic data selection is materialized by a novel and sophisticated technique called the *Hierarchical Virtual Node* which opens an application to interact with data directly in parallel coordinates that would otherwise have been impossible or difficult to achieve by existing methods. It works well even under the circumstance of the curse of dimensionality and offers several advantages over others. For example, the use case only requires a mouse click to select a set of data item(s). To achieve an efficient visual analytics, a set of analytic tasks are also developed in each layer of the proposed framework.

Chapter 1 Introduction

Information visualization is a broad field of study. The main area of interest primarily focuses on the optimal organization of abstract data into a visual representation with interactivity for knowledge discovery. The term knowledge discovery [1] refers to previously unknown and potentially useful information from the given data. Multidimensional visualization [2] is an important subfield of information visualization with primary interest in the study of multidimensional dataset, organized in a $n \times p$ matrix for n observations on p variables.

Thanks to Moore's law [3]. The rapid development of communication and storage technologies have enabled data to be ubiquitously collected at an unprecedented rate over the past few decades. The growing complexity of information space has posed many challenges to visualization especially for these application domains require decision making from a high dimensionality of data. These challenges urge a trend of moving towards the integration of analytic tasks with a higher degree of interactivity.

1.1 From InfoVis to Visual Analytics

In the book "*Graphics of large datasets: visualizing a million*" authored by Unwin et al. [4] with many techniques to present massive volumes of data were put forward. Successful mapping of data into a graphic representation, however, does not always imply the gaining of data insight. Kosara [5] asserted that if a visualization significantly increases the cognitive process of a learning then it is merely expressed in a form of visual art (see Figure 1.1). Mayer et al. [6] also pointed out that effective visualization must focus on reasoning. The key point is that visualization shall be practical and the information it reveals must be meaningful.

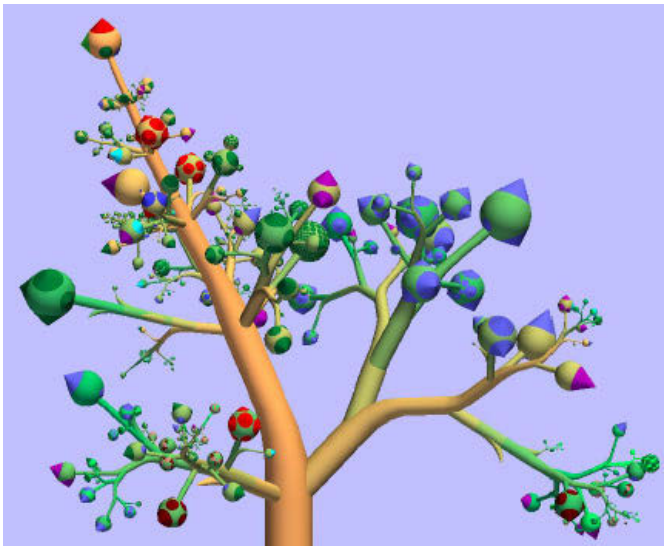


Figure 1.1. A botanic visualization of a hard disk, practical or art? The image is sourced from [7].

In 2005, van Wijk [8] gave a work titled "*The value of visualization*" where he used the study of his student [7] as an example (see Figure 1.1) to question that if a visualization failed to convey knowledge from its visual representation then it is merely functioned as an art. However, one of the key concepts in his study is the formulation of *a simple model of visualization* as depicted in Figure 1.2. It shows that the knowledge is derived from perception via an interactive exploration to produce a useful image. Keim et al. [9] further developed a sense-making loop based on it and the relevant phrase is quoted as follow:

…… The solution offered by Visual Analytics is then to let the user enter into a loop where data can be interactively manipulated to help gain insight both on the data and the representation itself.

Manifestly, both tried to emphasize that the iterative interaction (main loop) with visualization is the key element in the lifecycle of a visual analytics.

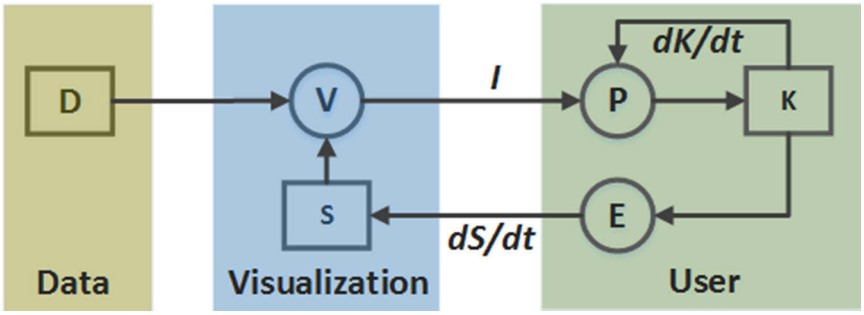


Figure 1.2. A simple model of visualization. The image is reproduced based on [8] where D, V, S, P, K, E, and I denote data, visualization, specification, perception, knowledge, exploration and image. dK/dt is the amount of knowledge gained and dS/dt means the interactive exploration by adapting a specification to a visualization.

In the taxonomy of visualization techniques contributed by Keim [10], many techniques developed to be efficient are finding themselves no longer adequate to meet the analytical needs without further integrating data mining, statistics, machine learning or other reasoning. This inadequacy has resulted in the demand for an effective framework by covering relevant theories collectively to deal with data complexity. Visual analytics [11] emerges as an important field by introducing interdisciplinary dependencies across scientific fields as illustrated in Figure 1.3. Thomas [12] defines the term visual analytics in his book "*Illuminating the Path*" as *the science of analytical reasoning facilitated by interactive visual interfaces*. Overall, the objective of interdisciplinary integration is to provide an effective framework for gaining the perception and knowledge and eventually making a decision from a complex structure of data that would otherwise have been impossible to achieve by a standalone field.

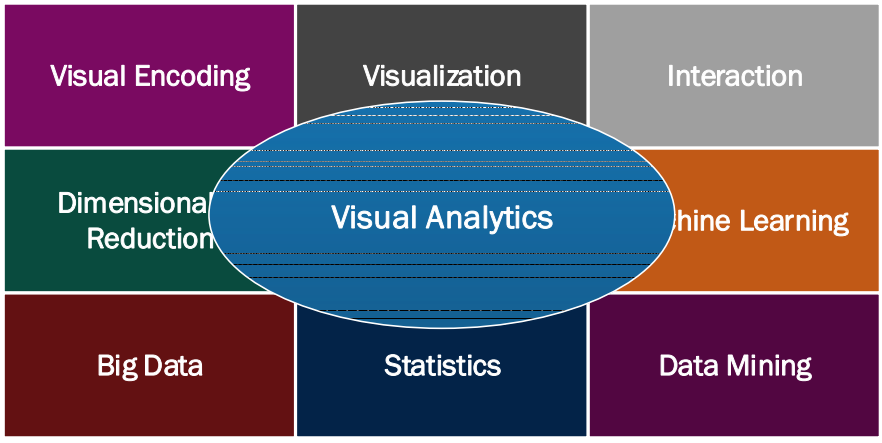


Figure 1.3 Visual analytics is an integration of interdisciplinary theories. The image is reproduced based on [11].

1.1.1 Problem Statement

The term "*interaction*" is becoming overloaded (a.k.a a buzzword) in information visualization. In almost all the cases, every software tool is interactive through a widget (such as a data grid, list box and etc.) which is cumbersome. Strictly speaking, a visualization is just one software application. There is no difference whatsoever to other software applications. What makes a visualization really special is how well its interactive component can help the user to carry out visual analytics in an intuitive, efficient and user friendly manner. This is indeed a significant problem if one further considers overplot and visual clutter in multidimensional space. Mathematically, if one models visualization as a *super* function then the problems can be classified into inputs and output. For example, a visualization deals with data (input parameters) and applies analytical reasoning (input command) to generate an image (output). Too many visualizations claiming to be interactive have skipped inputs and only focused on output. The details of how to accept inputs in an efficient, accurate and direct (not via widget) manner to an application have often been neglected. Perhaps, this is because they assume the traditional interaction of a widget style.

The aim of this research is how to improve the interaction mechanism in multidimensional visualization such as in parallel coordinates visualization. The existing interaction approaches that are currently used in the parallel coordinates [13] visualization cannot perform the 'Select' operation in Yi's [14] seven-layer interaction model. Unlike graph visualization in which a 'Select' operation can be easily achieved through a mouse-click (or mouse-rollover) on a shaped geometric region (a node), there is no geometric region allocated to polylines in parallel coordinates geometry. Therefore, a mouse-click (and mouse-rollover) operation over a particular polyline (visual object) is impossible. Consequently, the 'Select' interaction in parallel coordinates visualization is also theoretically impossible.

However, in visual analytics, data selections and data retrievals are very common operations. Without these operations, a multidimensional visualization can only be used as a data viewing tool. It cannot be deeply involved in the data analytics process.

1.2 Challenges and Goals

For challenges in visual interaction, Thomas [12], in his book "*Illuminating the Path*" claimed that:

Visual representations alone cannot satisfy analytics needs. Interaction techniques are required to support the dialogue between the analyst and the data. ... more sophisticated interactions are also needed to support the analytics reasoning process. ...

The key phrase here is the *sophisticated interaction* which implies novel and non-trivial. While the point based interaction design is successfully applied in graph visualization for supporting analytics reasoning, it is still in its very preliminary stage in terms of applying in multidimensional visualization. This is because most of multidimensional visualization techniques are based on polyline data representation, which does not occupy a geometrical region for supporting point based interactions. Therefore, some analytics reasoning processes are difficult to be implemented in multidimensional visualizations. Overall, the goals of this dissertation are:

- investigate new interaction techniques that can support analytics reasoning directly in multidimensional visualizations,
- investigate new visual data selection and data retrieval techniques through direct point based interactions on polyline based data visual representations,
- apply a set of analytic reasoning algorithms into our proposed interactive visualization to evaluate the effectiveness and efficiency of new approaches in terms of how well our approaches can support analytics reasoning processes, and
- materialize a multidimensional visualization system that tightly integrates developed data selection and a set of analytics reasoning tasks.

1.3 Contributions

In summary, the contributions of this dissertation include but are not limited to:

1. A new framework of visual interaction in multidimensional visualization (Chapter 3).

2. A novel and interactive data retrieval (or data selection) technique called the Hierarchical Virtual Node (HVN) approach in parallel coordinates visualization (Chapter 4).

Other additional contributions are:

3. A divide-and-conquer model developed on the basis of our new framework of visual interaction for dealing with a big dataset (Section 5.2.4).

4. A successful application of Rough Set Theory (RST) for dimensionality reduction in multidimensional visualization for visual data analytics (Section 5.3.1).

5. An enhanced scatterplot matrix method for visual data analytics (Section 8.1).

6. A new space filling multidimensional visualization (SPMDVis) (Section 8.2).

1.4 Outline

This dissertation is structured as follows:

- Chapter 2 Background: Covers an overview of existing visualizations and interactive techniques.
- Chapter 3 A New Framework of Visual Interaction: Proposes a new model of visual interaction based on existing frameworks.
- Chapter 4 Hierarchical Virtual Node: A complete chapter is dedicated to describing the technique and the implementation of the hierarchical virtual node.

- Chapter 5 Interactive Techniques for Visual Analytics: This chapter introduces a set of analytic tasks for visual analytics by further expanding the model described in Chapter 3. The core technique of interaction for the tasks is based on the hierarchical virtual node described in Chapter 4.
- Chapter 6 Technical Evaluations: Presents the technical evaluations of our developed system based on the HVN against other publicly available visualization systems.
- Chapter 7 Case Studies: This chapter presents three case studies for the applications of the techniques described.
- Chapter 8 Extended Works: This chapter introduces two extended works about the scatterplot matrix and a new space filling multidimensional visualization.
- Chapter 9 Conclusion: This chapter concludes the dissertation.

Chapter 2 Background

In this chapter, we provide an overview of the multidimensional visualizations and interactive techniques related to this dissertation. The concept of the *curse of dimensionality* is also explained under subsection of terminology which describes the problem domain that is commonly encountered when dealing with multidimensional data.

2.1 Terminology

We The author will kindly refer to himself and the contributing parties collectively as *we*. This is to sincerely acknowledge the contributions of that others have made towards the completion of this dissertation.

Dimension A dimension holds a data vector and is commonly referred to as *variable* or *attribute* in many scientific disciplines so these terms will be used interchangeably throughout this dissertation.

Multidimensional Data A dataset with arbitrary number of dimensions N and M observations. For simplicity, it is usually organized in a form of matrix $X = M \times N$.

$$X = \begin{pmatrix} X_{11} & \cdots & X_{\mathrm{M1}} \\ \vdots & \ddots & \vdots \\ X_{\mathrm{1N}} & \cdots & X_{MN} \end{pmatrix}$$

We will use D and P to denote the column vector and row vector respectively. A data vector D contains M observations for a variable X_i and a data row P holds only one observation for N dimensions. In other words, the notation of $D_i = \{d_1, d_2, \dots, d_M\}, \forall d \in X_i$ and $P_i = \{d_1, d_2, \dots, d_M\}, \exists! \, d_{ij} \in X_j$ refer to a column and row major vector respectively.

Data Element A data sample denotes as d_i in an univariate data vector D. The term *data element* is frequently referred to as *data item* or *data point* so these terms will be used interchangeably throughout this dissertation.

Object-oriented In the description of an algorithm, we often use dot notation to imply the access to a field or method of an object Obj for simplicity. For example, $Obj.HasMoreChildren$ or $Obj.HasMoreChildren()$ means the access to an object's field and method respectively.

2.1.1 Curse of Dimensionality

The study conducted by Lyman et al. [15] in 2003, estimated the information digitally stored had grown nearly 30 percent between 1999 and 2002. A decade later, of course, the trend is still continuing at a rate faster than ever in the age of *big data*. Information collected with multiple attributes such that $X = \{a_1, a_2, \dots, a_n\}$ is known as multidimensional data. High dimensionality creates extra complexities upon existing challenges by orders of magnitude. To name a few, it needs more space for storing the data and more time for searching the spare feature as well as more training data for learning in classification. These phenomena are broadly known as the *curse of dimensionality* which is the term first coined by Bellman [16] to describe the growing complexity of the problem in terms of solving nonlinear optimization in dynamic programming with high dimensionality.

Multidimensional visualizations inherit the curse of dimensionality as more dimensions bring more challenges. The growing complexity of the visualization depends on the increase in dimensionality, scale of data and non-linearity of the dimensions. An example is illustrated in Figure 2.1 showing that the application of bar chart, line chart and parallel coordinates to visualize one, two and multi-dimensional data respectively. Obviously, the complications start to rise gradually from low to multi-dimensional visualizations, making it more difficult to understand the meaning and more time consuming to interpret the result from the visualization.

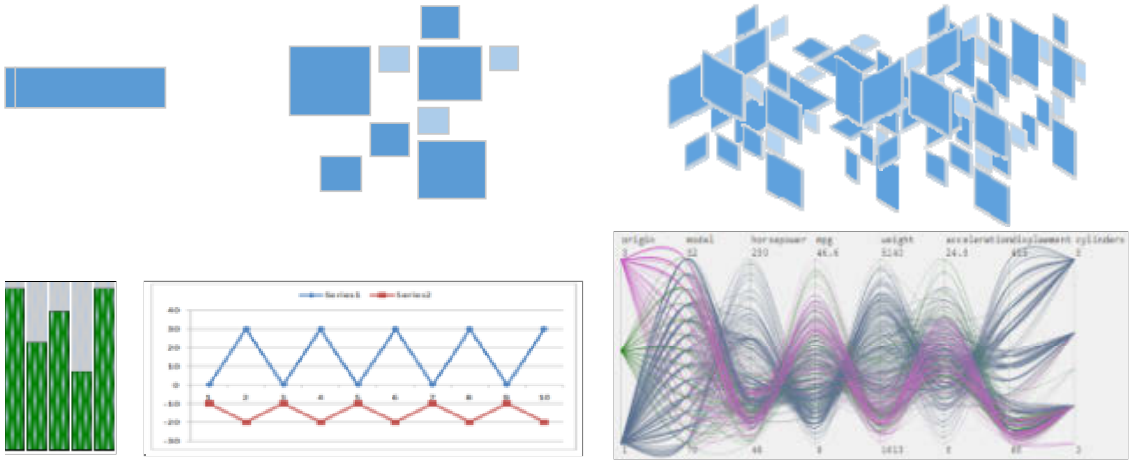


Figure 2.1. Illustration of visual and data complexities.

In summary, the curse of dimensionality will reflect in a visualization as follows:

1. Decline in visual perception because the phenomena of interest are often sparse in multidimensional space. For example, it is easy to perceive the data distribution and its linearity in a scatterplot rather than in parallel coordinate.
2. Create visual clutter and over-plotting.
3. Decline in learning accuracy due to data noise because not necessarily all the variables ought to be analyzed. For example, in multiple regression, it is often a time consuming task to analyzed all the variables first and then remove those variables with less contribution (in terms of statistical significance) from the model.
4. Increase prediction error and as a result, adds to the cost of interpretation. For example, when classifying more variables would more training data in order to improve the classification rate.

The understanding of the curse of dimensionality is important because in its attempt to deal data with the high dimensionality of data it preempts problem domains that commonly exist for all sciences.

2.2 Multidimensional Visualizations

In this section, we will review multidimensional visualizations on the basis of the taxonomies of visualization techniques presented by Keim [10] [17] and Kerigel [17]. Basically, they have classified the techniques into:

- Pixel oriented.
- Geometric projection.
- Icon based.
- Hierarchical based, and
- Graph based.

In particular, we will focus on the categories of pixel oriented, geometric projection and hierarchically based visualizations. The visualizations reviewed will be technical and comprehensive especially, for parallel coordinates because our interactive framework has been developed on the basis of it.

2.2.1 Parallel Coordinates

The origin of parallel coordinates is vague. It is often believed that it was proposed by Maurice d'Ocagne [18] in the 19th century. Strangely, the book written by d'Ocagne is mathematical and has no connection with the parallel coordinates visualization. However, in the mid-20th century, Inselberg [13] brought it back into awareness. Nowadays, it is probably the most well-known and extensively used multidimensional visualization. This is evident in a Google search with an illustration in Figure 2.2 where the results returned for the keyword "*parallel coordinates*" is about 26.63 times more than the keyword "*scatteplot matrix*".

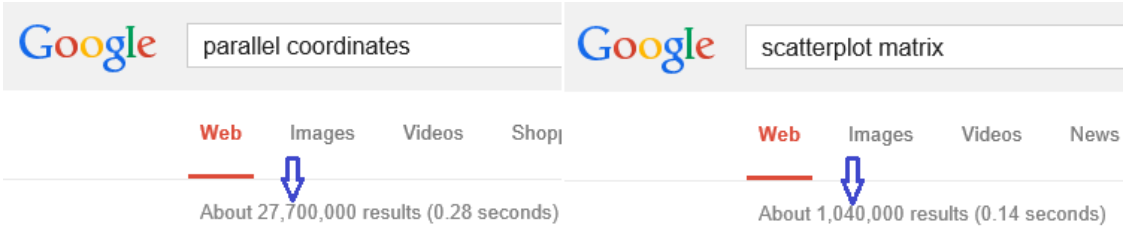


Figure 2.2. A comparison of the Google search results. Google returned 27,700,000 and 1,040,000 results for the keywords "*parallel coordinates*" and "*scatterplot matrix*" respectively.

Parallel coordinates is considered to be a geometric projection based technique. Given a set of variables $X = \{X_1, X_2, \dots, X_N\}$ in which each X_i is drawn as a vertical axis across a horizontal plane. The vertical axes serve the end knots of a polyline denoted as $P_i = \{d_1, d_2, \dots, d_N\}$ and every data point in P_i is associated with one and only one variable such that $\exists! \, d_i : X_i$. We assume that the origin of the screen coordinate starts from the bottom-left corner[1] on the target platform. It is trivial to compute the y-coordinate of a data point d_i with respect to its variable X_i by the equation below.

$$y_{d_i} = Y_{X_i} + \left(\left(\frac{d_i - min_{X_i}}{max_{X_i} - min_{X_i}} \right) \times height_{X_i} \right)$$

Equation 2.1

Where Y_{X_i}, $height_{X_i}$, min_{X_i} and max_{X_i} are the y-coordinate of a vertical axis, height of the vertical axis, minimal value and maximal data value with respect to X_i. The output y_{d_i} is a transformed value of d_i in screen coordinate. These notations are straightforward. Please refer to Figure 2.3 for clarity.

[1] Typically, the 2D GUI coordinate system starts from the top-left corner and the origin of 3D starts from the bottom-left.

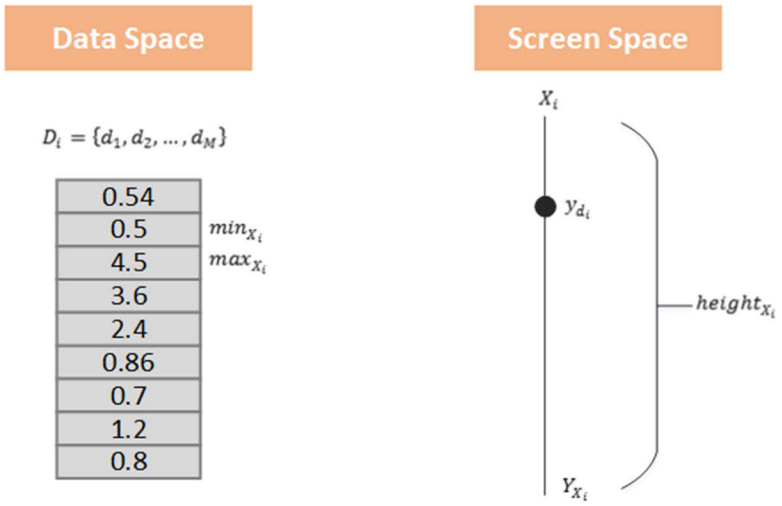


Figure 2.3. Mapping a data point to a vertical axis. The diagram describes the notations and how a data point is mapped to the screen coordinate.

The key property of a polyline is that it establishes the integrity of a multidimensional data item geometrically. To perceive the data pattern, one needs to discern a set of polylines with similar undulation.

Algorithm 2.1 provides an implementation of parallel coordinates where the details of $DrawPolyLine$ and $DrawCircle$ have been skipped because they are simply the wrapper functions of the software API. For example, GDI in the Windows platform.

---

1. **procedure** *DrawParallelCoordinate*(X, ω, ν, y)
2. $X = \{X_1, X_2, \ldots, X_N\}$
3. ω /* Constant line width of a vertical axis. */
4. ν /* Constant height of a vertical axis. */
5. $startY = y$ /* Starting y-coordinate of a vertical axis. */
6. **Initialization:**
7. $P = \emptyset$
8. $startX = 0$
9. $offset = screenWidth \,/\, (N - 1)$
10.
11. /* Draw the vertical axis here. */
12. **for** $i \coloneqq 0$ **to** N

---

13. $endY = startY + v$
14. $DrawLine(startX, startY, startX, endY)$
15. $startX = (startX + offset)$
16. **end for**
17.
18. /* Draw the geometric primitives here. */
19. **for each** X_i **in** X
20. **for each** d_i **in** D_{X_i}
21. $y_i = yPointToScreen(d_i, max_{X_i}, min_{X_i}, v)$
22. $x_i = X_{X_i}$ /* A data point always attaches to a vertical axis, i.e. variable. */
23. $P \leftarrow (x_i, y_i)$ /* Add the coordinate to the list. */
24. $DrawCircle(x_i, y_i)$
25. **end for**
26. $DrawPolyLine(P)$
27. **end for**
28. **end procedure**

Algorithm 2.1 An implementation of the parallel coordinates visualization.

In Algorithm 2.1 where D_{X_i} holds a column vector with respect to X_i and $yPointToScreen$ is a wrapper function of Equation 2.1. X_{X_i} denotes the x-coordinate of a vertical axis X_i. Note that, there is no need to map a data point to the x-coordinate which is constant and always equal to its respective X_{X_i} so only the computation of the y-coordinate is necessary.

Figure 2.4 shows the implementation result of Algorithm 2.1 with an application of car dataset[2]. The view has been optimized to promote the location proximity of correlated variables while maximizing patterns.

[2] Car dataset has obtained from StatLib, Carnegie Mellon University, see http://lib.stat.cmu.edu/datasets/.

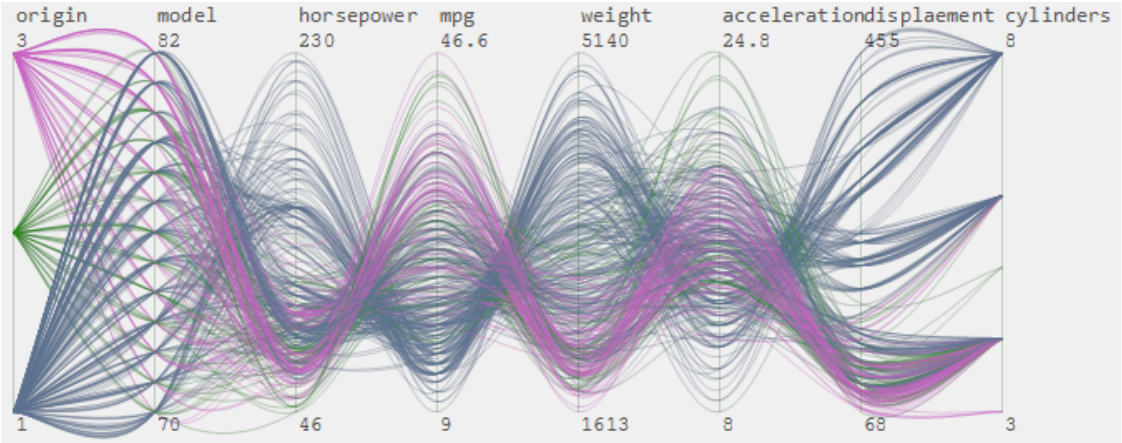


Figure 2.4. A parallel coordinates visualization. Car dataset is used and the geometric primitive is based on Bezier curve. The techniques applied here are brushing and dimensional reordering.

Parallel coordinates is extensively used to discover data patterns but its visual effectiveness is greatly dependent on the spatial arrangement of the variables. For example, a subtle change in permutation can lead to a totally disparate pattern (polyline undulation) than the others. One can compare Figure 2.4 with Figure 2.5 which used the default ordering. Even though, a brushing technique has been applied in Figure 2.5 but the view overall is more disorganized than the one produced in Figure 2.4.

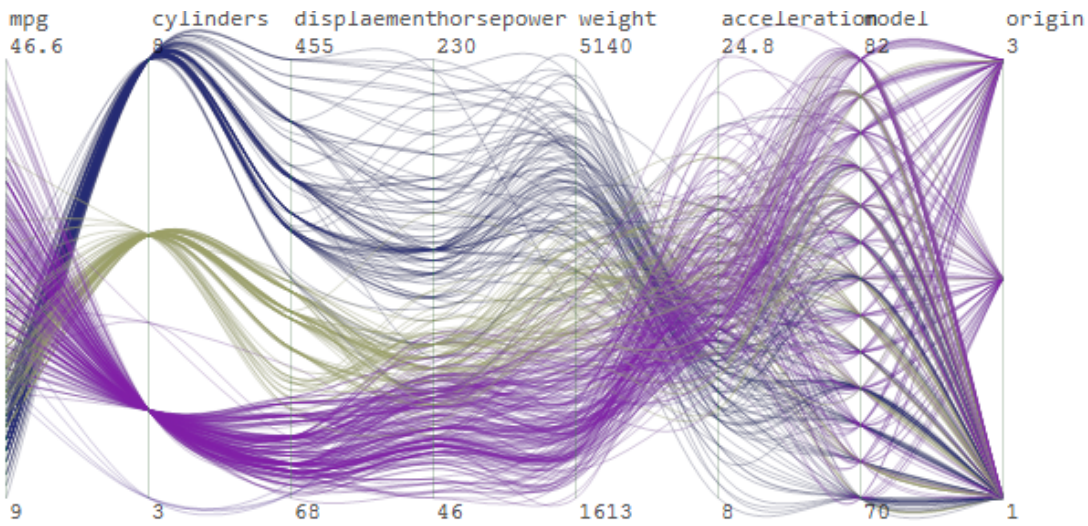


Figure 2.5. A parallel coordinates visualization in default variable ordering. The dataset used in this example is the same as per Figure 2.4.

Misinterpretation of Linear Correlation The most confusing part of parallel coordinates is the interpretation of the polyline slope. From the perspective of human cognition, the slope intuitively suggests a linear relationship but this is not entirely true. The coordinate system of parallel coordinates is not based on the Cartesian system which uses the perpendicular line $\perp XY$ to divide a plane into four quadrants. Instead, parallel coordinates projects data only in one direction so the concept of slope $y = mx + c$, where $m = \frac{\Delta Y}{\Delta X}$ is really not applicable. This explanation aims to clarify that parallel coordinates is not suitable to be used to discern a linear correlation in the way that scatterplot is capable of.

A visual perception of the linear correlations between the parallel coordinates and scatterplot matrix is provided in Figure 2.6. Here, one can easily perceive that there might exist a linear dependency for a data pattern (data subset between pairwise variables) in parallel coordinates but the overall correlation revealed in scatterplot suggests a different interpretation. For example, it is hard to imagine that a) has no correlation, b) is negative correlated and c) is less correlated or nonlinear. Especially, the patterns between b) and c) are subtle in parallel coordinates, but scatterplot suggests a totally different trend.

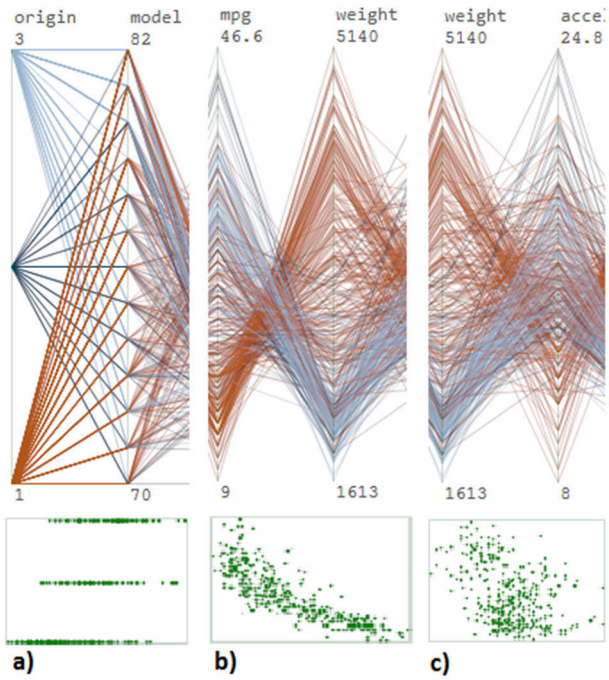


Figure 2.6. The perception of linear correlations between parallel coordinates and scatterplot matrix. Car dataset is used again. (Top) Snippet of the parallel coordinates. Pairwise variables are$\{origin, model\}$, $\{mpg, weight\}$and $\{weight, acceleration\}$ from left to right. (Bottom) Scatterplot for the corresponding variables above.

2.2.2 Scatterplot Matrix

Scatterplot matrix [19] is widely used in statistics for multivariate exploratory data analysis. It is conceptually simple and should be considered as an extension of the classic scatterplot [20] rather than an independent subject. It is classified as a geometric projection technique by Keim and Kerigel [17].

Given a set of variables $X_1, X_2, \dots, X_i$, a pairwise variables $X_i, X_j, i \neq j$ is plotted where X_i and X_j are also known as independent (horizontal axis) or dependent (vertical axis) variable respectively. Since $\{X_j, X_i\}$ is a transposed plot of $\{X_i, X_j\}^T$ and an identity plot $\{X_i, X_i\}$ is essentially a 45° line for a continuous or 180° line for discrete variable. Therefore, it is widely acceptable to display either full or tri-diagonal matrix. The implementation of the scatterplot matrix is trivial and Equation 2.1 can be reused by invoking it twice, one for the y-coordinate and another call for the x-coordinate. Figure 2.7 shows the visualization of the full scatterplot matrix where the lower triangular matrix is essentially a transposed view of the upper triangular matrix and vice versa.

The scatterplot matrix presents multidimensional data in such a way that it enables the perception of linear correlations over an entire dataset simultaneously. On the other hand, the major disadvantage is the linear reduction of screen space allocated to each scatterplot. For example, let *M*, *P*, *K* denote scatterplot size, entire display size and number of variables respectively. The size of a scatterplot can be trivially computed as $M = {}^{P}/_{K}$ where *M* declines rapidly when *K* increases and eventually, the visualization will become a point cloud. In general, parallel coordinates is more space efficient than the scatterplot matrix.

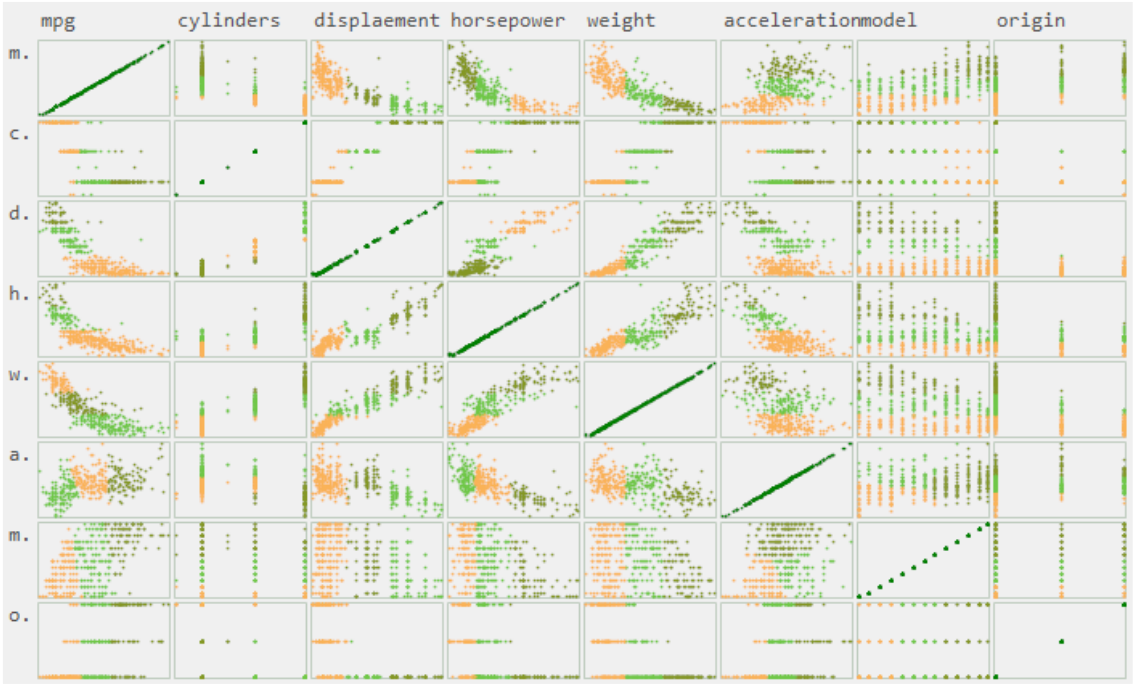


Figure 2.7. A visualization of the scatterplot matrix.

2.2.3 TableLens

TableLens [21] is a visualization for exploring a large amount of tabular information by merging graphical and symbolic representations into an interactive view. The visualization is tightly integrated with focus+context and zooming techniques. An important feature of TableLens is that the scaling of a view is independent of each other in either the horizontal or vertical order. Figure 2.8 provides an illustration of the TableLens visualization. According to the taxonomy by Keim and Kerigel [17], TableLens is considered as a geometric projection based technique.

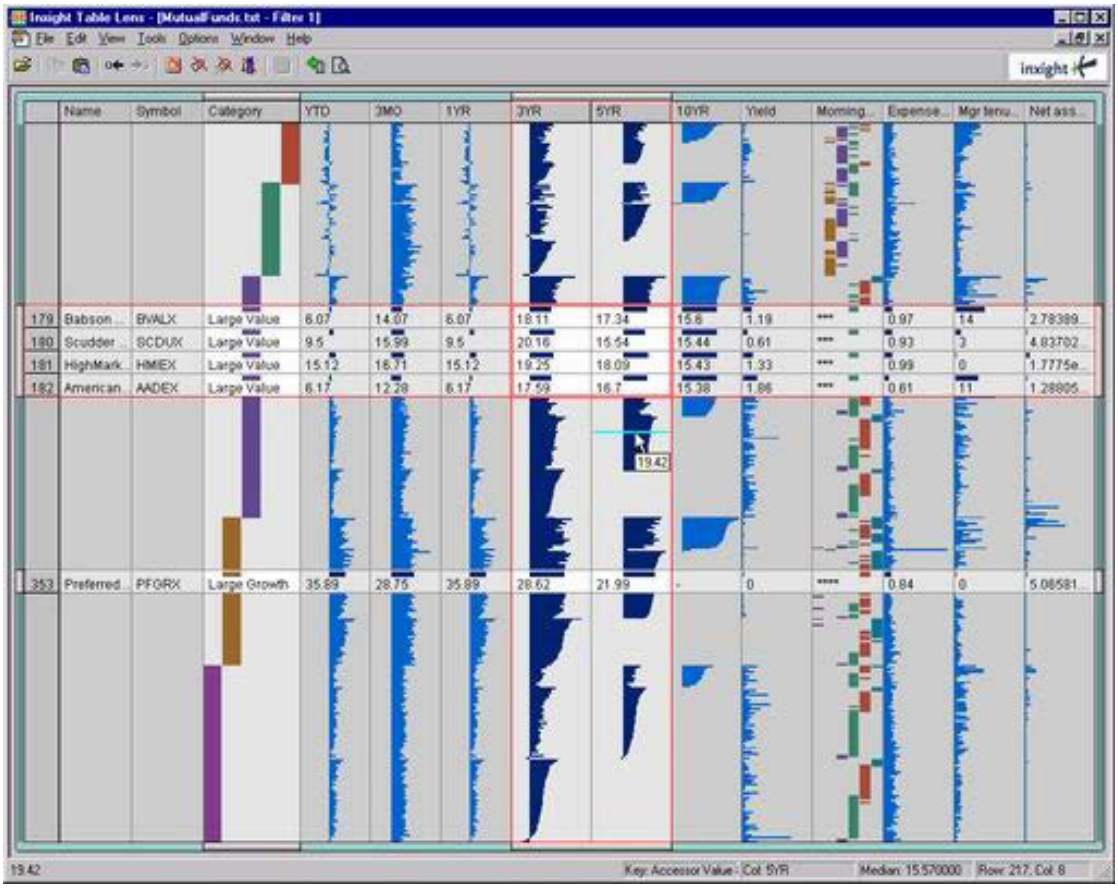


Figure 2.8. The TableLens visualization. The image is sourced from http://www.ramanarao.com/articles/2001-12-online-info/cviz.html.

2.2.4 Space Filling Curve

The Hilbert curve [22] was first described by David Hilbert [23] in the 19th century. It is a continuous and self-similar space filling curve with many useful applications such as spatial database indexing or mapping of high dimensional data into lower dimensional space such as multidimensional indexing. It is considered as a mixture of geometric and pixel oriented techniques.

The basic building block of a Hilbert curve is a one side opened rectangle which we call it a *Hilbert curve element*. Conceptually, the drawing process of the Hilbert curve is simple, one can imagine the entire plane is logically divided into a $N \times N$ grid and a grid cell is further partitioned into 4 quadrants. Each quadrant can be visited at most once. The points in the grid cells are connected to form an element as shown in Figure 2.9 (Left).

Finally, self-similar neighboring elements are connected together to form a continuous Hilbert curve in Figure 2.9 (Right).

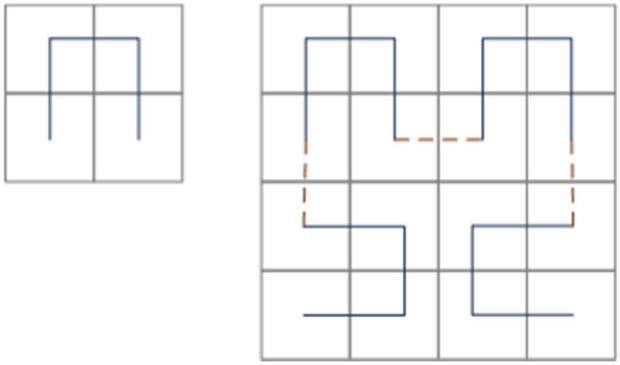


Figure 2.9. Building block of Hilbert curve. (Left) A single element of a Hilbert curve. (Right) A space filling curve in a 8×8 cells. An orange dotted line indicates the join with the other element.

The guiding operations can be encoded by three symbols [24] $\{F, +, -\}$, interpreted as "move forward", "turn 90° to the left" and "turn 90° to the right". Therefore, the representation $+F - F$ instructs the algorithm to *move forward after turning* 90° *to the left and then move forward after turning* 90° *to the right*.

Figure 2.10 provides the visualizations of the Hilbert curve in different orders. For a more advanced application, elements are usually coded by various colors to denote the spatial separation.

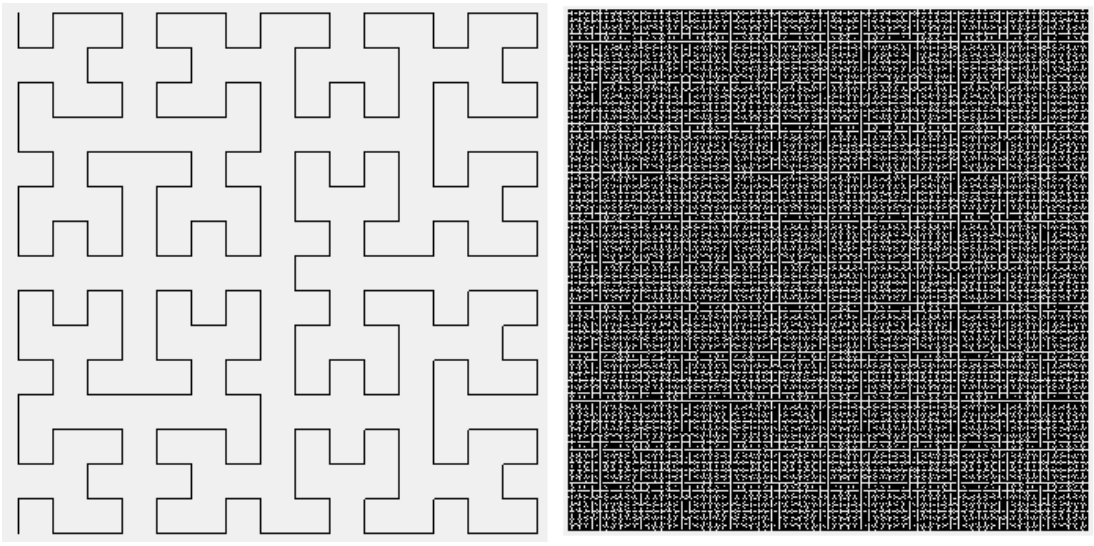


Figure 2.10. Visualizations of Hilbert space filling curve. The depth of recursion is 4 and 8 for left and right diagrams respectively.

2.2.5 Star Coordinates

Star coordinates (SC) [25] is a geometric projection based multidimensional visualization that arranges axes on a circle in such a way that every axis starts at the same origin. The coordinate system of SC is curvilinear where the data can be transformed into a Cartesian coordinate by summing all the unit vectors in each coordinate and multiplying by the data value that is similar to Equation 2.1 as defined for parallel coordinates.

Star coordinates visualization presents multidimensional data in a way similar to the scatterplot matrix. For example, both were designed on the basis of scatterplot with the main difference being the coordinate system but the overall approach of perceiving data is similar. Figure 2.11 illustrates a star coordinates visualization.

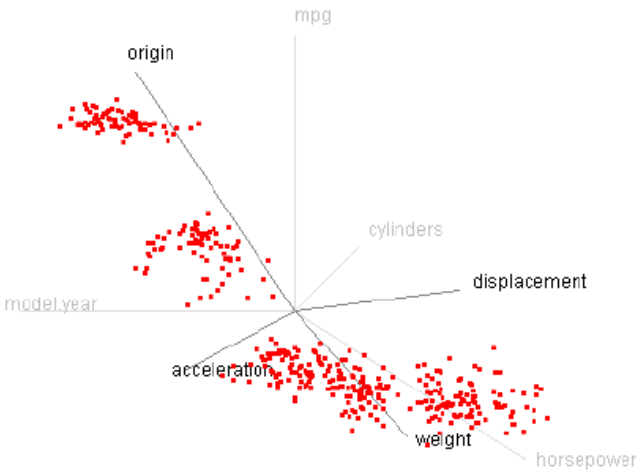


Figure 2.11. Star coordinates visualization. The image is sourced from [25].

2.2.6 TreeMap

TreeMap mainly deals with hierarchical data. Normally, the discussion of TreeMap shall not be mixed with multidimensional visualization but Cao et al. [26] has successfully

designed a TreeMap-like interface to visualize multidimensional clusters so it will be briefly introduced here.

TreeMap is a hierarchical based visualization that was first introduced by Shneiderman [27] to tackle the problem of visualizing hierarchical data such as a file system structure. The TreeMap visualization has been successfully commercialized and a renowned application in the real world is probably the *Map of the Market* that was developed by SmartMoney[3] [28].

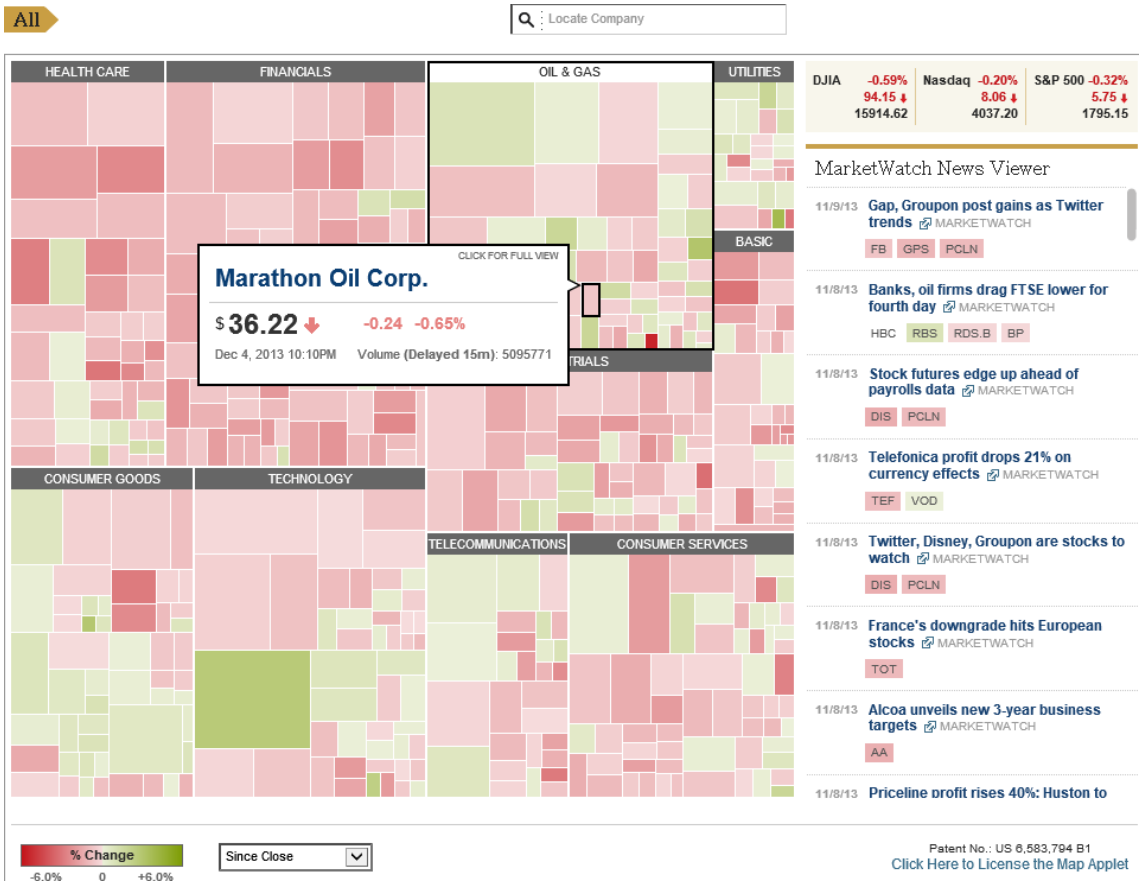


Figure 2.12. Map of the market. The tool shows the live market data in TreeMap. The diagram is sourced from the MarketWatch website (http://www.marketwatch.com).

[3] Dow Jones ceased SmartMoney magazine. The September 2012 issue of SmartMoney was the magazine's last edition. All the contents and tools from SmartMoney are available on MarketWatch.com

The key feature of the TreeMap is the ability to fully utilize display space. Given a hierarchical dataset, the layout algorithm starts off with one rectangle that occupies the entire display initially and is latter divided into nested rectangles recursively while the algorithm traverses down along hierarchical data paths. This process continues until the bottom of the hierarchy has been reached. Figure 2.13 illustrates the conceptual mapping of the hierarchical data to the rectangles in the TreeMap.

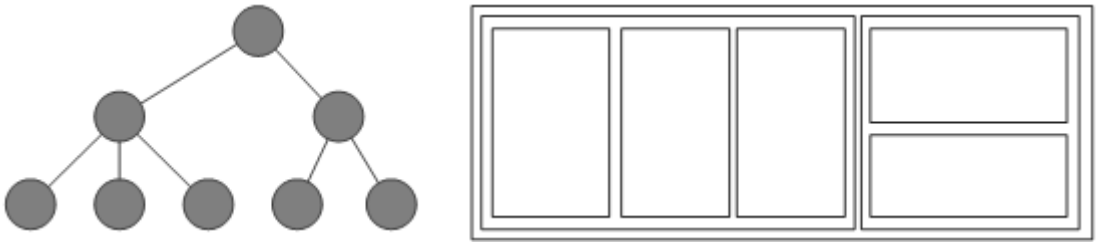


Figure 2.13. Hierarchical data mapping.

There are variety of TreeMaps [27] [29] [30] but the layout principally remains similar and in general, an optimal layout algorithm tries to produce rectangles with the aspect ratio as close to one as possible.

The slice and dice TreeMap [27] is the first and simplest TreeMap layout algorithm as it recursively divides a rectangle into rectangles using parallel lines. Sub-rectangles represent children to its parent rectangle.

The squarified TreeMap [29] is another variation of the TreeMap that works by dividing rectangles in horizontal and vertical rows. A rectangle is either added to the current row or the current row is fixed and a new row is started. The decision to determine whether a rectangle shall be fixed or continue its subdivision is given by the following function:

$$worst(R, \omega) = \max_{r \in R}\{\max(\omega^2 r^+ / s^2), s^2/(\omega^2 r^-)\}$$

Where r^+ and r^- denote the maximum and minimum value of R and the width is given by ω.

2.3 Interaction in Multidimensional Visualization

This section will review the common tasks of interaction with multidimensional visualization. Usually in a visual analytics, one interacts with visualization for data retrieval, and views change or analytic reasoning so they can be grouped for easy understanding.

2.3.1 Data Retrieval

Data retrieval (a.k.a data selection) refers to the process of expressing the interested subset of data for application by a subsequent task.

Widget based data selection This approach allows users to interact with data indirectly through the traditional user interface such as tabular display as illustrated in Figure 2.14. Data are usually presented in their raw form with no (or little) information to describe their characteristics. Thus, the user needs to be familiar with the underlying dataset for meaningful data selection in such a raw format.

The greatest advantage is its simplicity because very little effort needs to be expanded to bring about rapid prototyping. However, the disadvantages include the lack of visual indication and also the fact that data are usually organized in a natural order so the effort to locate interested items can be quite significant.

CPS_85_Wages.xml

education	south	sex	experience	union	wage	age	race
8	0	1	21	0	5.1	35	2
9	0	1	42	0	4.95	57	3
12	0	0	1	0	6.67	19	3
12	0	0	4	0	4	22	3
12	0	0	17	0	7.5	35	3
13	0	0	9	1	13.07	28	3
10	1	0	27	0	4.45	43	3
12	0	0	9	0	19.47	27	3
16	0	0	11	0	13.28	33	3
12	0	0	9	0	8.75	27	3
12	0	0	17	1	11.35	35	3
12	0	0	19	1	11.5	37	3
8	1	0	27	0	6.5	41	3
9	1	0	30	1	6.25	45	3
9	1	0	29	0	19.98	44	3
12	0	0	37	0	7.3	55	3
7	1	0	44	0	8	57	3
12	0	0	26	1	22.2	44	3
11	0	0	16	0	3.65	33	3
12	0	0	33	0	20.55	51	3

Figure 2.14. Widget based data selection. The user interacts with raw data rather than the visualized data. It is difficult to select data subset without understanding the nature of the dataset.

Direct point selection This technique is often seen in graph or nodal based visualizations such as a scatterplot in Figure 2.15. A node occupies a concrete region in the display with the duality of representing coordinates and data points simultaneously. Therefore, the common use case of data selection in such visualization allows users to select data directly in the display. In general, point based selection is intuitive but its availability is limited to nodal based visualizations.

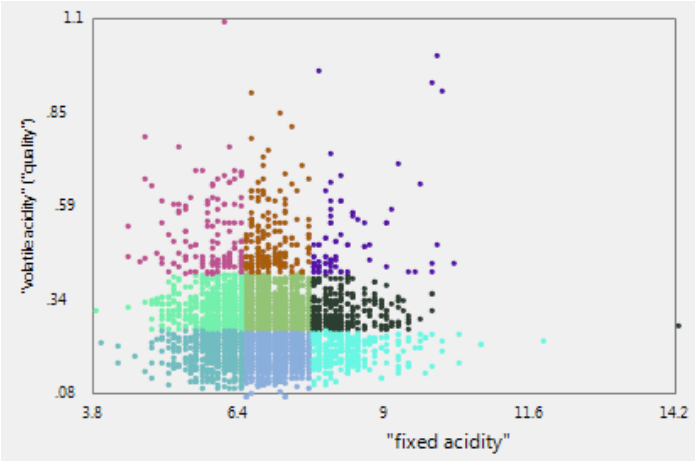


Figure 2.15. An example of scatterplot. The scatterplot typically uses point selection technique by allowing user to interact with a data point by a mouse clicking.

Direct 2D data selection It allows users to select a set of data on a 2D plane. This technique will be reviewed again comprehensively in Section 4.2. For such an application in multidimensional visualization, Siirtola et al. [31] has applied a similar technique in parallel coordinates. Basically, it selects a collection of polylines passing through the rectangular area which is drawn by a sequence of mouse click, drag and release operations. Obviously, it does not work well over a visualization with intensively overlapping elements. Indeed, Siirtola has commented that it is more appropriate to use 2D selection for highlighting outliers.

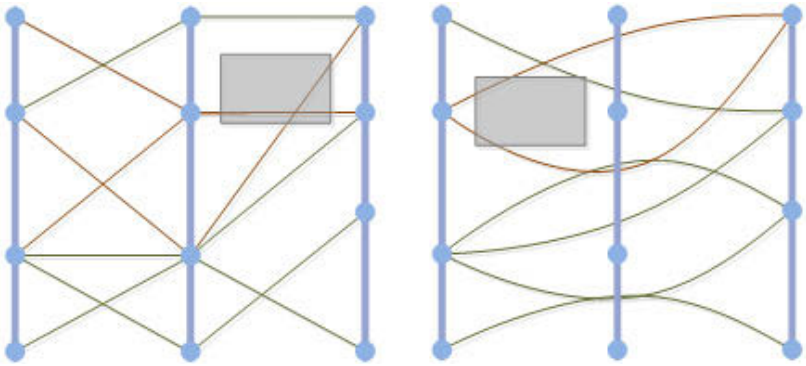


Figure 2.16 Direct data selection by a 2D rectangle. This method is used to select a set of data but its accuracy tends to decline in a crowded visualization.

2.3.2 Interaction for View Change

Brushing By using a brushing interaction, a subset of data items can be highlighted (or focused) for viewing the detail of these data patterns. This technique is widely used in parallel coordinates. In 2002, Hauser et al. [32] proposed the concept of angular brushing as an extension to the standard brushing to facilitate the data subsets grouping and highlighting the rational data properties of the date items. They also integrated the composite brushing and focus+context technique to further improve the visual exploration in parallel coordinates.

In addition, Zhou et al. [33] discussed the visual clustering technique which allows users to specify transfer functions in order to control the density value of the lines using alpha blending. We consider the visual clustering as a variation of brushing. One significant drawback of applying alpha blending in brushing is that the low density patterns tend to become illegible due to high transparency. This is certainly not desirable if the low density pattern is statistically significant. Overall, brushing remains a rudimentary technique which is popular mostly due to its simplicity.

Filtering It is a fundamental technique of data manipulation that attempts to minimize data noise for uncovering interested data in a crowded visualization. In a visual analytics, a noise typically refers to unwanted data with respect to a current task. In parallel coordinates visualization, a good example of filtering is probably the system implemented by Artero et al. [34]. They introduced an interactive filtering method by first computing the frequency and density information. Such information is subsequently used to filter out the data for greater visual perception for clutter reduction. Filtering requires better familiarity with the analyzed dataset otherwise users can potentially filter out some meaningful patterns or even create a poor view.

The major difference between *filtering* and *brushing* is the output strategy of highlighting and grouping of user expected (interested) data items. The former hides or dims the less important (or less interested) data items and the latter displays the complete dataset and sets the unique color to the selected data item in order to differentiate a selection from the rest.

Zooming It is generally concerned with the level of abstraction. A conceptual illustration of zooming technique is provided in Figure 2.17. Basically, we consider zooming as a general term that covers classic zooming, focus+context and detail-on-demand. In multidimensional visualizations, TableLens and a system implemented by Fua et al. [35] are good examples. Fua [35] discussed the applications of using drill-down, roll-up and dimension zooming techniques for navigating the level of detail in parallel coordinates

and the techniques developed have been further integrated into XmdvTool[4] [36] since version 3.1.

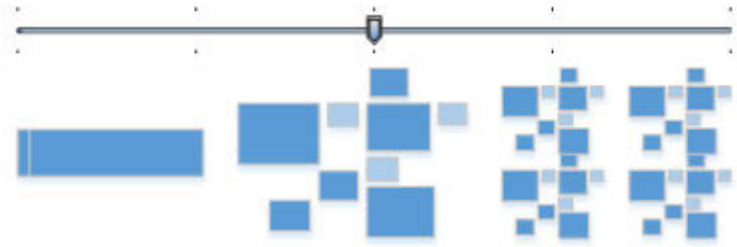


Figure 2.17 An illustration of zooming technique. The diagram illustrates the applications of zooming to abstract a dataset into different levels of perspective.

For focus+context, Novotny and Hauser [37] discussed an interesting work for outlier detection. The basic idea is to put the detail into focus while preserving the relations to other data that is also known as the context. In other words, this idea is similar to independent dimension scaling by varying the scale of one or few variables while fixing others simultaneously.

Dimensional Reordering The technique is widely used in parallel coordinates visualization. Recall that parallel coordinates is mainly used to explore data patterns but that does not necessary imply patterns will be divulged naturally. The overall geometric structures presented by parallel coordinates are susceptible to the ordering of variables and inappropriate ordering tends to create visual clutter due to tangled line crossing. This technique aims to promote the location proximity of correlated variables for uniform undulation.

There are numerous studies about the technique of variable reordering in parallel coordinates. For example, Ankerst et al. [38] developed a technique to arrange dimensions based on the similarity measurement. Peng et al. [39] used dimension reordering to rearrange variable axes based on their visual neighbouring similarity for clutter reduction.

---

4 Current version of XmdvTool is 8.0 released on October 20, 2010.

Yang et al. [40] further contributed a technique based on optimal and heuristic ordering. Furthermore, Huang et al. [41] presented a classification based method to maximize the uniform undulation of geometric primitives and the result is shown in Figure 2.18.

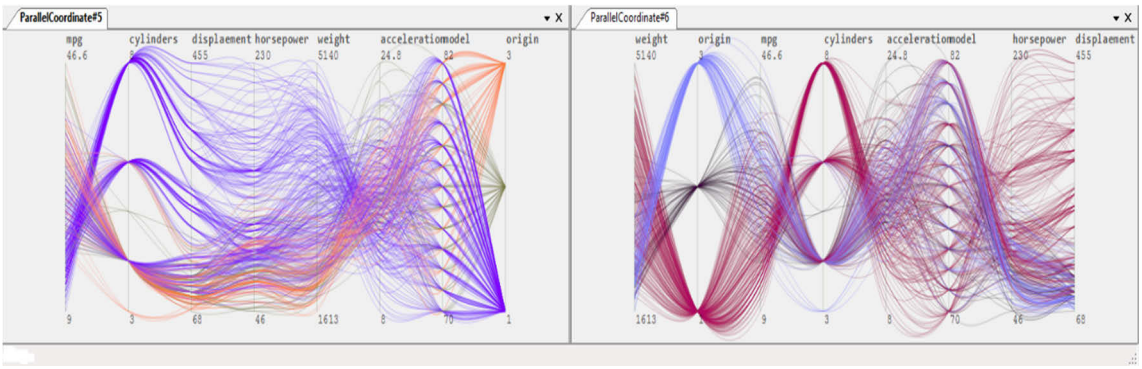


Figure 2.18 Reordering of variables in parallel coordinates. (Left) Parallel coordinates visualization before dimension reordering. (Right) Parallel coordinates visualization after dimension reordering.

2.3.3 Interaction for Analytical Reasoning

2.3.3.1 Clustering

Clustering techniques are often used in visual analytics to cluster data into groups on the basis of a statistics principle rather than an arbitrary selection.

K-means [42] [43] is a simple method that aims to partition data into K clusters. The algorithm first initializes a set of K clusters $C = \{C_1, C_2, \dots, C_N\}$ by the random selection of K data points to be the centroids accordingly. Each data point is assigned to a nearest cluster C_i with the following equation.

$$\sum_{i=0}^{K} \sum_{\forall d \in C_j} \|d - \mu_i\|^2$$

Equation 2.2

Where μ_i and d denote the centroid and a data point in C_i respectively. The centroid is updated in each iteration and the process continues until there is a convergence. That

is, there is no change or the change can be tolerated. The efficiency of K-means is largely determined by the speed of convergence and a heuristic is often used to select a good initial data point for quick convergence. The drawback of K-means is the likelihood of a convergence to a local optimum so it cannot guarantee that the outcome is always globally optimum.

Figure 2.19 demonstrates our application of K-means where each cluster has been color brushed for the visual separation of clusters and the convex hull algorithm is used for plotting the boundary.

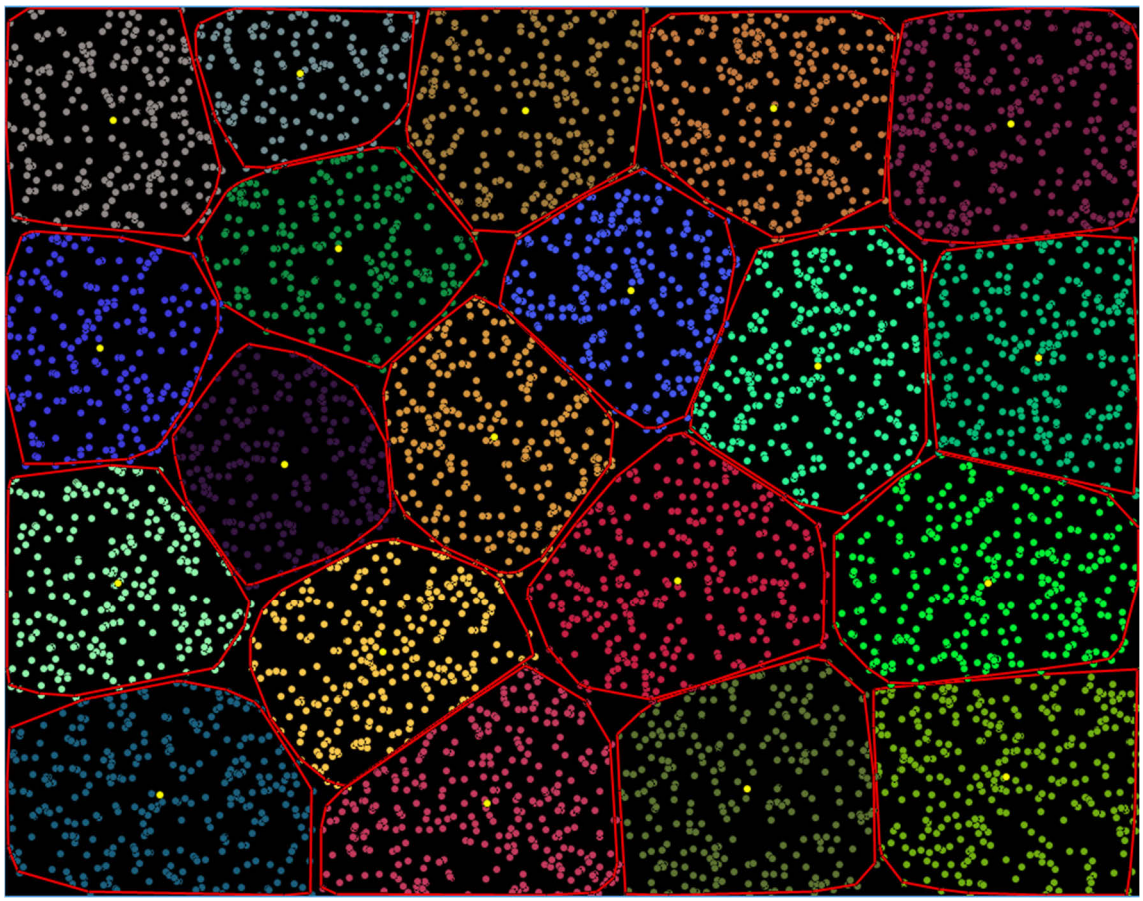


Figure 2.19. A K-means clustering. The randomly generated data have been partitioned into 20 clusters where the yellow dot indicates the centroid of a cluster.

Hierarchical Clustering [44] It is a non-parametric technique and probably one of the most widely used clustering methods in many scientific applications. Given a set of data, the method partitions them into a set of hierarchically disjointed clusters in the following form.

$$G = \{g_1 \cup g_2, \dots \cup g_N\}^1, \dots, \{\{g_1 \cup g_2\} \cup \{\dots\}, \dots \cup \{g_N\}\}^{\delta} : g_i \cap g_j = \emptyset, i \neq j$$

Where δ denotes the depth of the hierarchy. The hierarchy of the clusters are built iteratively by merging two clusters with an objective function to form a new cluster in each iteration. Generally, there are two categories of hierarchical clustering namely, agglomerative and divisive. They are bottom up and top down for the former and the latter respectively. A conceptual illustration is provided in Figure 2.20

Unlike K-means, it does not require K clusters to be known in advance but it needs a stopping rule to terminate the process when an optimal number of clusters have been found. Hierarchical clustering is used in our proposed technique of interactive data selection and will be reviewed comprehensively in Chapter 4.

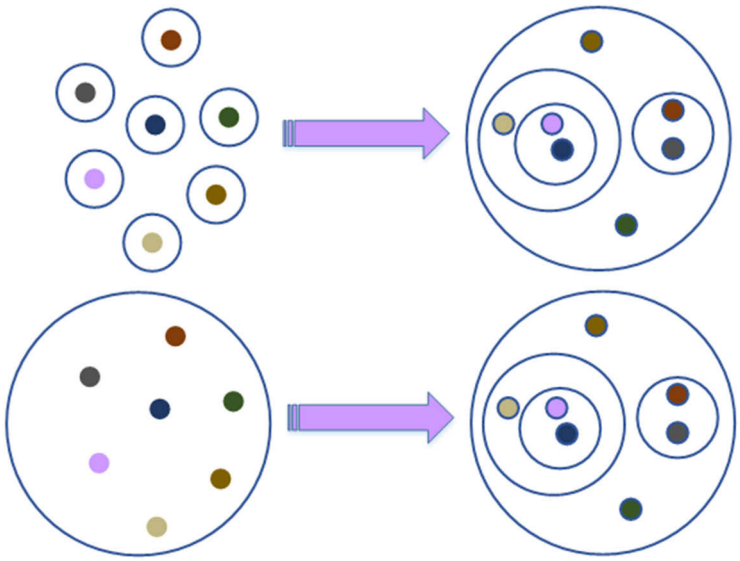


Figure 2.20. Hierarchical clustering categories. (Top) Agglomerative. (Bottom) Divisive.

2.3.3.2 Dimensionality Reduction

Dimensionality reduction is an advanced analytic task and is commonly used in many sciences to attenuate the curse of dimensionality. The basic principle is to map a high dimensional dataset $X = \{X_1, X_2, \dots, X_N\}$ into a lower information subspace $X = A \cup B, A \cap B = \emptyset$ while preserving the original interestingness. The terms *interestingness* is subjective and is therefore dependent on the objective of an algorithm. Let A and B denote the reduced and excluded subset respectively such that $\forall A \in X, \forall B \in X$. The

reduced subset A shall be sufficient to describe the original superset X. There are two classes of dimensionality reduction namely, supervised and unsupervised. In short, supervised methods allow users to influence the outcome by a set of parameters and vice versa.

Visual Hierarchical Dimensional Reduction Yang et al. [45] contributed an automatic and manual brushing mechanism to parallel coordinates in their work called Visual Hierarchical Dimensionality Reduction (VHDR). VHDR has been integrated into XmdvTool [36] since version 6.0. The interaction technique offered by Yang is capable of exploring a large dataset in a more interactive manner with greater flexibility to dynamically change the view. VHDR first constructs a hierarchy of a dimensional tree grouped by similarity and further allows users to interactively select an interested dimensional cluster for analysis.

User-Defined Combinations of Quality Metrics Johansson et al. [46] introduced a supervised method of dimensionality reduction in the field of visualization called the User-Defined combinations of Quality Metrics (UD-QM). They used a set of metrics such as Pearson correlation, outlier and cluster detection to rank variables. In UD-QM, the prerequisite knowledge required to quantify the quality metric parameters might need greater user expertise. For example, users need to define the correlation, outlier and cluster values in such a way as to avoid insignificant correlations, outliers and clusters adding up to a sum that appears to be significant.

The techniques described above were proposed primarily in the field of information visualization. In addition, there are many well-known methods of dimensionality reduction proposed in statistics and for a good taxonomy, one can refer to [47].

Principal Component Analysis It is often abbreviated by the acronym PCA. Mathematically, it performs an orthogonal linear transformation by mapping data to a lower dimensional space with non-trivial computation of covariance matrix and eigen-problems. There are two commonly used selection criterions to select principal components namely, the Kaiser criterion and the Scree test. For adopting PCA in the

dimensionality reduction, the Kaiser criterion [48] is perhaps the widely acceptable criterion by ignoring the components with eigenvalues less than one. The Scree test is another popular criterion which was proposed by Cattell [49] who suggested plotting the eigenvalues on the graph to find a smooth decrease then cutting off the line to retain those components appearing on the left hand side of the cut point. For example, Figure 2.21 illustrates the use of Scree test to reduce a dataset from 8 to 2 variables. The disadvantage of using PCA or other unsupervised methods is the unexpected outcome because the operation was carried out without any consideration of user inputs and hence the unexpectedness is often criticized as an information loss.

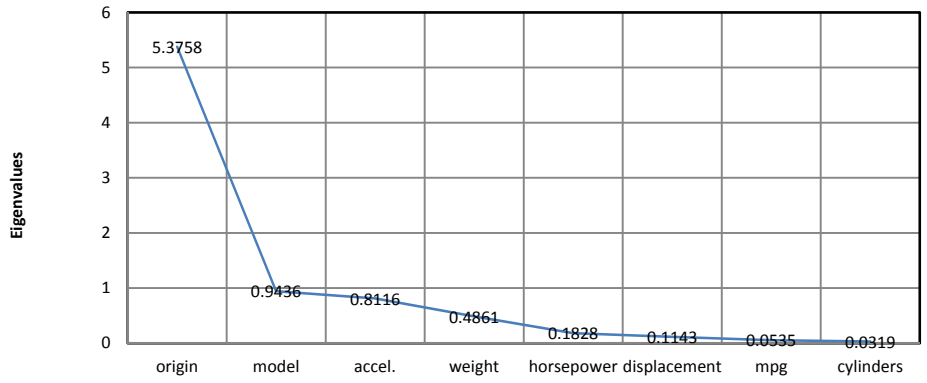


Figure 2.21 A plot of eigenvalues for the Scree test.

Projection Pursuit (PP) [50] is a linear method to pursue the choices of possible projections that can reveal the interested structure defined by a projection index. To pursue the possible projections globally involves a task of non-trivial computation, as described in Huber's [51] study. XGobi [52] had already integrated PP for viewing high dimensional data. The main problem of PP is the difficulty to quantize the value of the projection index since it is possible to present interested structures spuriously given an inappropriate projection index.

Rough Set Theory (RST) The rough set theory was first introduced by Pawlak [53] in the field of approximation to classify objects in a set and in general it is applicable to any problems that require classification tasks. Given a dataset, let U be a finite set of objects

called the universe and A be the superset of all attributes $A = \{a_1, a_2, a_3, \ldots\ldots a_N\}, \exists!\, a_i \in A$ such that $a: U \rightarrow V_a,\ \forall a \in A$ where V_a is called the domain of a. A is further classified into two disjoint attribute subsets D called the decision attribute and rest the condition attributes C such that $A = (C, A \cup \{D\}), C \cap D = \emptyset$. For any objects $X \in U$ with non-empty subset $P \subseteq C \cup D$ are said to be discernible with respect to P if and only if the following equivalence relation is true.

$$f_P(X_i, X_j) = \begin{cases} 1, if\ V_a(X_i) = V_a(X_j), \forall V_a \in P, for\ i \neq j \\ 0, otherwise \end{cases}$$

Equation 2.3

Clearly, given the equivalence relation defined in Equation 2.3, one can construct equivalence classes denoted as $U/IND(P) = \{E_1, E_2, \ldots, E_n\}$ by partitioning U into disjoint subsets with the following indiscernibility relation.

$$IND(P) = \{(X_i, X_j) \in U : f_P(X_i, X_j) = 1\}$$

RST further defines three regions of approximation called the lower approximation $\underline{P}X$, upper approximation $\overline{P}X$ and boundary region to approximate subsets $X \subseteq U$. The lower approximation and upper approximation are also called the positive and negative region respectively. The lower approximation contains objects that are securely in X and the upper approximation consists of objects that cannot be classified to X whereas the boundary region contains objects that possibly belong to X.

The RST is chosen in our system for the task of dimensionality reduction. The most distinct advantage of applying RST as a supervised method is the concept of *condition* and *decision*. Users simply specify a variable as decision and the rest become conditions so the variables are reduced in such a way that they fully respects user specified decision.

2.4 Discussion

Widget based style is the simplest way to interact with visualization and remains the most frequently used method. This can be understood because direct interaction in multidimensional visualization is very challenging due to the curse of the dimensionality.

Furthermore, in an empirical evaluation of various multidimensional visualizations, we use parallel coordinates as the main metaphor for our framework and interactive

techniques. There are many reasons for this. First, it is space efficient when compared with the scatterplot matrix. Second, it is relatively easy to understand and interpret multidimensional data as oppose to others such the space filling curve, TreeMap or TableLens. The overview of the multidimensional dataset can be completely visualized in single view whereas the scatterplot matrix uses multiple scatterplots to puzzle the overview of the multidimensional dataset.

Chapter 3 A New Framework of Visual Interaction

In this chapter, we present a new framework of visual interaction based on 7 layers framework proposed by Yi et al. [14] earlier. The proposed framework simplifies 7 layers into 3 layers for better understanding of the interactive tasks in multidimensional visualization.

3.1 Introduction

Interaction mechanism extends the capability of a visualization beyond a static image. From the perspective of cognitive science, Norman [54] pointed out that human beings are social organisms so it is natural for us to interact with others for knowledge (or message) transmission and interaction forms a fundamental aspect of our behaviour. In a more recent study, Liu and Stasko [55] presented a work explaining the relationship between mental model, visual reason and interaction from the view point of information visualization. Although, they came from different fields they coincided on the point that interaction plays a key role in human cognition. Figure 3.1 shows a conceptual model for the progressive development of knowledge discovery. The interaction with visualization to derive insight is an iterative process and each iteration refines a hypothesis while improving one's understanding towards the underlying data.

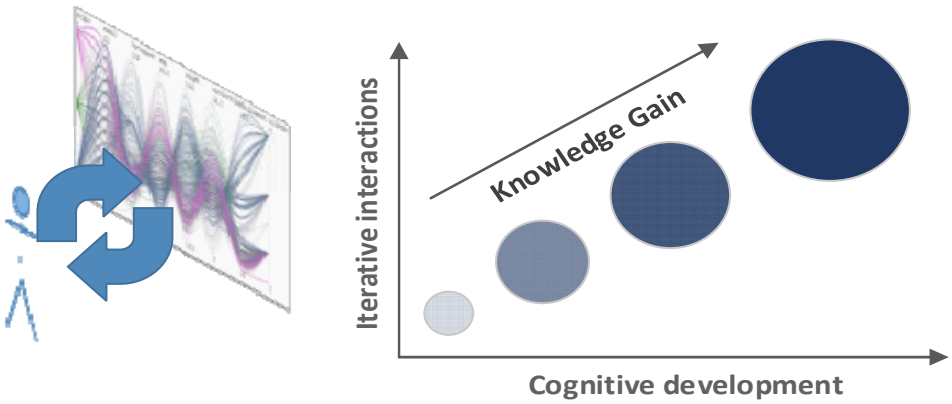


Figure 3.1 A cognitive model of gaining data insight.

There are diversities of interactive techniques already proposed in information visualization and the easiest way to gain a good understanding is to review the relevant works of taxonomy. In 1996, Shneiderman [56] provided a great taxonomy of interactive techniques classified by task types. There are seven abstract tasks defined as follows:

- Overview: Global view of the entire dataset.
- Zooming: Zoom in or out on the interested data.
- Filter: Remove the data noise.
- Details-on-demand: Present the details when needed.
- Relate: View relationship between two selections.
- History: Keep a history of operations for undo, redo and etc.
- Extract: Allow extraction of sub-collections.

In addition to the taxonomy contributed by Shneiderman. Yi et al. [14] argued that there exist many taxonomies but the discussions are often made from a low level operation's point of view. Hence, they proposed a taxonomy from the perspective of user's intent. That is, the tasks are classified from a user's intention rather than the nature of a task itself. Yi's model of visual interaction consists of 7 layers as follows:

- Select: Make something as interesting.
- Explore: Show me something else.
- Reconfigure: Show me a different arrangement.
- Encode: Show me a different representation.
- Abstract/Elaborate: Show me more or less detail.
- Filter: Show me something conditionally.
- Connect: Show me related items.

The '*select*' operation is used for highlighting or subset selection via the user interface. On the other hand, the '*explore*' operation is intended for finding out user-

interested data items through the visual navigation of a data source. The layers 3 to 5 concern the strategies of view change for better understanding and highlighting one (or more) portion(s) (or patterns) of the visualization that are currently perceived as interesting by the user. The last two layers use the '*filtering'* mechanism to display (or visualize) only the interested or related data items in the visualization and remove other less interested and related data items from the visualization. Certainly, there are still many excellent taxonomies such as [57] [58] not being covered here, but we believe that these two are particularly representative.

3.2 3-Layers Framework of Visual Interaction

Overall, there are many layers that overlap to some extent in Yi's model. To make easy understanding and better interpretation of the layered structure of visual interactions, we further propose a new model by refining Yi et al.'s [14] 7 layers into 3 layers, broadly based on the nature of the operations as follow:

- Dynamic selection (or locating) of data items
- Dynamic viewing of data (visual structure)
- Dynamic scoping of data (data structure)

The following table provides a best effort of mapping between our model in Table 3.1 (Left) and the models described by Yi [14] and Shneiderman [56] in Table 3.1 (Middle) and Table 3.1 (Right) respectively.

Refined Model	**User intention**	**Task based**
Dynamic selection	Select	Extract
Dynamic viewing	Reconfigure	Zooming
	Encode	Overview
	Abstract/Elaborate	Details-on-demand
	Connect	Relate
	Explore	History

Dynamic scoping	Filter	Filter

Table 3.1. A mapping of taxonomies of interactive techniques.

3.2.1 Tasks by Dynamic Selection

Tasks in this layer are concerned with data subset retrieval that is similar to the *Select*; layer 1, defined by Yi et.al [14]. The layer of dynamic selection is the frontier of a visualization for providing a user with a mean to select or look up particular data item(s) of interest. Therefore, its practicability greatly influences the efficiency and quality of the subsequent task. Usually, the immediate task after a data selection is to apply a visual or data analytic technique on the data subset. Technically speaking, a data selection bi-divides the dataset logically into *selected* and *unselected* sets. A conceptual example is illustrated in Figure 3.2.

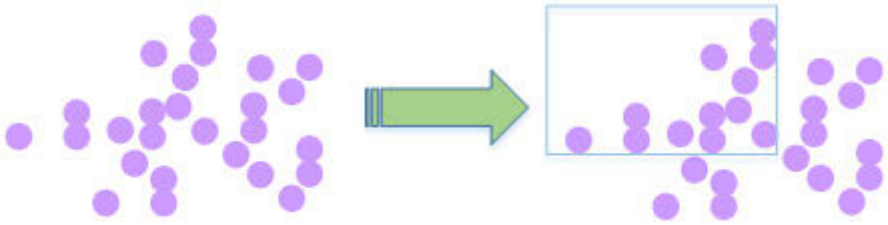


Figure 3.2. An example of dynamic selection operation.

3.2.2 Tasks by Dynamic Viewing

Dynamic viewing (DV) interaction, that merges layers 3, 4 and 5: *Reconfigure*, *Encode* and *Abstract/Elaborate* of the interaction defined in J. S. Yi's model, allows users to change data representations for achieving better readability or understanding of the data and its relational structures. Examples include the reordering of axes in parallel coordinates and navigation in the graph visualization by using a Hyperbolic Tree or a Fish Eye Browser. DV interaction also includes the change of visual encoding; that is using an alternative visualization method to present the same complete set of data.

In this layer, the primary concern is the interactive configuration of the visual aspect. A conceptual example is illustrated in Figure 3.3.

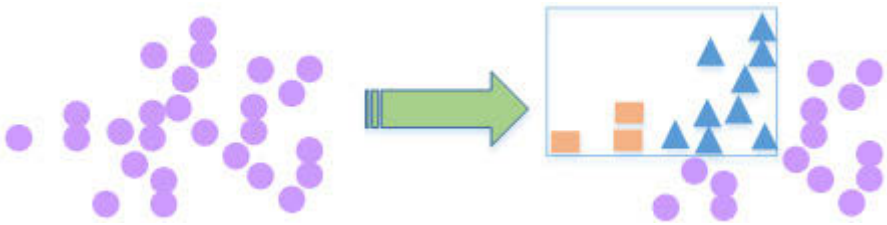


Figure 3.3. An example of dynamic viewing operation.

3.2.3 Tasks by Dynamic Scoping

Dynamic scoping (DS) interaction, that merges layers 2, 6 and 7: *Explore*, *Filter* and *Connect* of the interaction defined in J. S. Yi's model, allows users to visualize a subset of the data through the filtering of less important and relevant data items with respect to an analysis. Examples include the navigation method used in DA-TU [59]: an On-Line Visualization system and other dimensionality reduction techniques.

The effectiveness of data exploration has typically declined by a large number of dimensions. One of the motivations of dynamic scoping aims to shape the data to a smaller subset suitable for analysis while minimizing the visual clutter and information overloading. A conceptual example is illustrated in Figure 3.4.

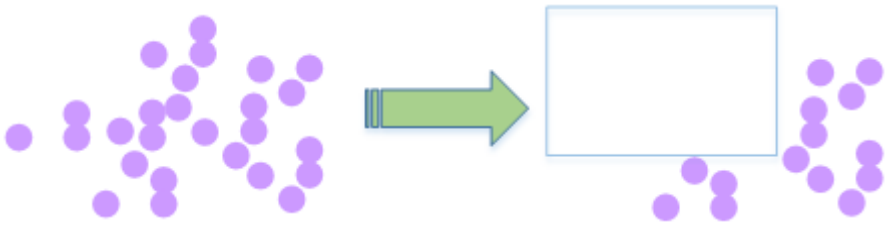


Figure 3.4. An example of dynamic scoping operation.

3.2.4 Discussion

Overall, the proposed framework of visual interaction tries to avoid the fine-grained classification because it is easier to understand an operation will result in a selection, visual change or data change. Overall, the proposed framework will serve as a design guideline for our interactive tasks to be discussed in the following chapters.

Chapter 4 Hierarchical Virtual Node

Data selection is the frontier of a visualization. The design goal is to translate a user's intention into a selection query via the designated interface. In the overall trend, data selection has received less attention in the development of interactive techniques since the term *interaction* is becoming overloaded. Basically, every software tool is interactive but mostly through the manipulation of a widget. Broadly speaking, a visualization is just one of the available software applications. What makes the interactive component of a visualization really distinctive is how well it supports arbitrary selection of data directly in the interface both intuitively and efficiently in order to facilitate subsequent analytic tasks. This is challenging especially in terms of interacting directly in multidimensional space due to the curse of dimensionality such as over-plotting, visual clutter etc.

In this chapter, a novel technique of *Hierarchical Virtual Node* (HVN) will be introduced which is revolutionary in such a way that it allows users to interact with data hierarchically, directly in parallel coordinates and even under the circumstances of over-plotting and visual clutter. However, the application of HVN does not limit itself only to parallel coordinates and is applicable to any visualizations with geometric primitives based on the polyline or polycurve.

4.1 Interaction or Selection?

Modern information visualization techniques, at their core, appear to have two main components: representation and interaction. The representation component is concerned with mapping from data to advanced computer graphics and how to draw or render them on the display. The interaction component on the other hand concerns the dialog between the user and the data stored on the system as the user explores the data set to uncover insights. The interaction component's roots lie in the area of Computer-Human Interaction (CHI). Although discussed as two separate components, representation and interaction clearly are not mutually exclusive.

While an information visualization system is taking the role of providing advanced GUIs for supporting Computer-Human Interaction (CHI), it is supposed to facilitate CHI in both directions; that is 1) the *input* from human to computer (or data), and 2) the *output* from computer (or data) to human. However, in the past decades, researchers in the InfoVis community have paid more attention to the *output* part; that is they are concerned more with the visual representation of output data, such as the output analysis results, for users to better understand its contents, attributes and relational structures. They have not paid enough attention to the human *input* part that is the human instructing, monitoring and guidance in the whole circle of visual data manipulation and visual analytical reasoning. The existing research work that has been done in the *visual human input* part has mainly focused on low-level zooming and navigation operations and has not addressed the benefit that human involvement provides in visual data manipulation and visual analytical reasoning processes.

4.2 Revisiting the Data Selection Models

First of all, we would like to revisit the models of data selection before going into the details of the HVN. In this section, numerous models, technical aspects and their awkward use cases of data selection will be thoroughly discussed. The technical provisions help one to understand that the complexity of the proposed HVN (see Figure 4.3) considerably surpasses others. Here, we aim to provide a comprehensive background about the current techniques and also help one to understand the significant contribution that the HVN has made and the problems that it aims to solve.

The following table presents a classification of the data selection models applied in parallel coordinates. The provision is on the basis of a courteous scan of the existing literature. Note that, the table is discretionary and by no means an exhaustive list since some authors did not explicit clarify the way to select data and also some systems are interactive only for zooming and, viewing rather than data selection. Therefore, we have decided to exclude them from the list.

Models	References

Rectangle Inclusion	[31]
Value Range	[60], [61], [62], [63]
Point Selection	[64][5], [65], [66]

Table 4.1. A taxonomy of data selection technique.

4.2.1 Rectangular Selection Model

The operations require one to draw a bounding region by the sequence of mouse down, drag and release operations. The bounding region defines the coordinates for searching the embraced data items. It has been widely used in graph visualization since a node has a duality of representing the data and coordinate simultaneously. For implementing this model in graph visualization, one has to test a point (X, Y) with the following conditions:

$$\begin{cases} rect_X \leq X \leq (rect_X + rect_{width}) \\ rect_Y \leq Y \leq (rect_Y + rect_{height}) \end{cases}$$

Please note that, these conditions presume the screen coordinate starts from bottom-left corner.

In the parallel coordinates, a geometric polyline or curve is technically drawn by passing multiple end points to a software API. Strictly speaking, a line other than both ends does not occupy a bounding region nor does it represent any data points. Thus, a practical implementation will need to test a slope-intercept between four sides of the rectangle against a given line. Figure 4.1 illustrates the operation of the rectangular selection model in parallel coordinates and scatterplot.

[5] Authors did not actually implement a point based selection model in their work. However, we have classified it here since its geometric primitive possibly allows a point based selection.

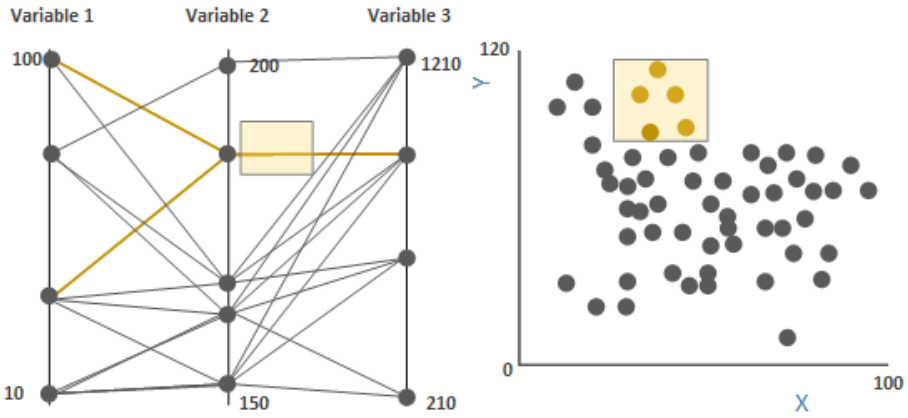


Figure 4.1 Rectangular selection model. (Left) A direct data selection in parallel coordinate, one has to find the intercept between line and rectangle. (Right) A direct data selection in scatterplot which is more accurate and intuitive.

For searching the line-rectangle interception efficiently the Liang-Barsky algorithm [67] is a desirable choice because it can return the occurrence an inception and also the coordinates of interception. Although, the interest here is to determine a line and rectangle interception one can further use the information of intercepted coordinates for focus+context operation within the rectangular area. The algorithm developed by Liang-Barsky was originally used to determine the interceptions between a line and its clipping window. It formalizes a line segment into parametric representations described below. Please refer to the diagram in Figure 4.2 for clarity.

$$x = x_0 + \Delta x \times t$$

$$y = y_0 + \Delta y \times t$$

Where Δ denotes a distance between two end points in one direction such that $\Delta x = x_1 - x_0$ and $\Delta y = y_1 - y_0$ and t is a parametric value which is 0 at the point (X, Y) and 1 at the point $(X + \Delta X, Y + \Delta Y)$.

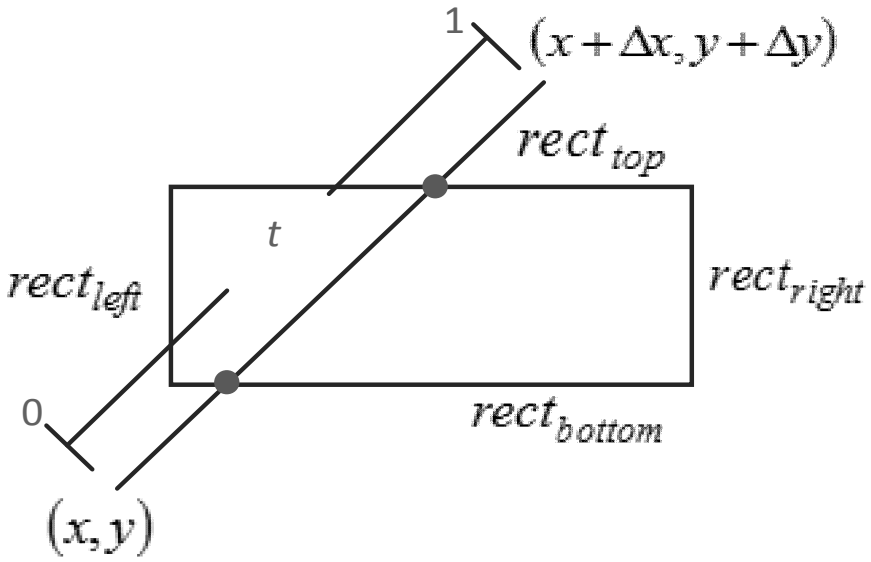


Figure 4.2 Properties of the Liang-Barsky algorithm. The diagram describes the notations and concept of the algorithm.

By substituting the parametric representations (see [67]), one can lead to the following notations.

$$\begin{cases} p_1 = -\Delta x \\ p_2 = \Delta x \\ p_3 = -\Delta y \\ p_4 = \Delta y \end{cases}$$

$$\begin{cases} q_1 = x_0 - rect_{left} \\ q_2 = rect_{right} - x_0 \\ q_3 = y_0 - rect_{bottom} \\ q_4 = rect_{top} - y_0 \end{cases}$$

The values $\{p_1, p_2, p_3, p_4\}$ and $\{q_1, q_2, q_3, q_4\}$ will be passed to the algorithm for testing the condition against the parametric value t. A versatile implementation is described in Algorithm 4.1 where $|*|$, X and D denote the cardinality, dimensions and data points. There are two inputs required namely P for a set of polylines and $rect$ which describes the attributes of the bounding rectangle.

1.	**input**
2.	$P = \{p_1, p_2, \dots, p_N\}$
3.	$rect$ /* Bounding rectangle. */
4.	**initialization**

5. $t_0 \leftarrow 0$
6. $t_1 \leftarrow 1$
7. $P \leftarrow \emptyset$
8. **begin algorithm**
9. **for** $i \coloneqq 0$ **to** $\|X\|$
10. $j = (i + 1)$
11. **for** $k \coloneqq 0$ **to** $\|D\|$
12. $x_0 = X_{X_i}$ /* X is the x-coordinate of X_i in here. */
13. $x_1 = X_{X_j}$ /* X is the x-coordinate of X_j in here. */
14. /* convert to screen coordinate with respect to X_i and X_j. */
15. $y_0 = yPointToScreen\big(D_i(k)\big)$
16. $y_1 = yPointToScreen\left(D_j(k)\right)$
17. **if** $LiangBarsky(-\Delta x, (x_0 - left_{rect}), t_0, t_1)$ **is** true **and**
18. $LiangBarsky(\Delta x, (right_{rect} - x_0), t_0, t_1)$ **is** true **and**
19. $LiangBarsky(-\Delta y, (y_0 - bottom_{rect}), t_0, t_1)$ **is** true **and**
20. $LiangBarsky(-\Delta y, (top_{rect} - y_0), t_0, t_1)$ **is** true **then**
21. $P \leftarrow D_i(k)$
22. $P \leftarrow D_j(k)$
23. **end if**
24. **end for**
25. **end for**
26. **return** P
27. **end algorithm**
28.
29. **procedure** $LiangBarsky(p, q, t_0, t_1)$
30. **if** $p = 0$ **and** $q < 0$ **then**
31. **return** false /* parallel line found. */
32. **else**
33. $r = {}^{p}/_{q}$
34. **if** $p < 0$ **then**

```
35.          if r > t1 then
36.              return false
37.          else if r > t0 then
38.              t0 ← r
39.          end if
40.      else if p > 0 then
41.          if r < t0 then
42.              return false
43.          else if r < t1 then
44.              t1 ← r
45.          end if
46.      end if
47.  end if
48.  return true
49.  end procedure
```

Algorithm 4.1 An implementation of the Liang-Barsky. The algorithm consists of two functions where the top function iterates through the lines in parallel coordinates and calls the function LiangBarsky which returns a Boolean to indicate whether the intercepted condition is true or not.

The main drawback is the noise within selected data tends to increase rapidly when parallel coordinates becomes cluttered. Under those circumstances, it is difficult to apply serious analytic techniques due to an unacceptable amount of unwanted data being included, as illustrated in Figure 4.3.

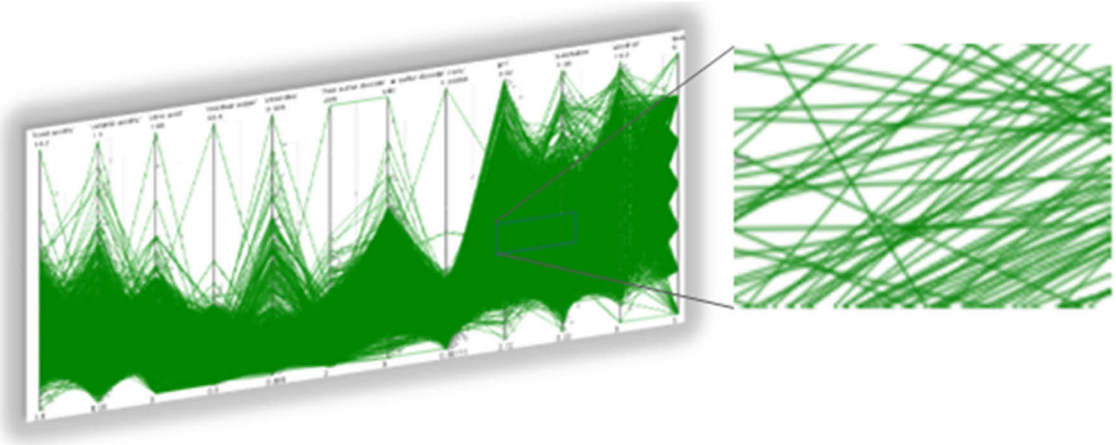


Figure 4.3. Rectangular selection model within a crowded visualization. It shows the challenge to select the intended data accurately when a visualization is overcrowded.

In summary, this model provides a rapid and agile data selection for a moderate amount of data but the user's frustration will soon arise when dealing with a crowded visualization due to inevitable inclusion of unwanted data. For example, the parallel coordinates system implemented by Siirtola [31] can select a collection of polylines which pass through the rectangular area but the use case does not work well over a display with intensively overlapping elements as Siirtola had further made a comment that it is more appropriate to harness rectangular inclusion for highlighting outliers.

4.2.2 Value Range Model

According to Table 4.1, in which this model appears to be the most popular in parallel coordinates probably due to its simplicity, there are numerous implementations but in general, users do not interact with geometric primitive directly. For example, they interact with widgets (i.e. a slider bar) attached to a vertical axis by adjusting its upper and lower value range. Let l_i and u_i be the inputs of upper and lower bound with respect to a target variable X_i, one can easily use a conditional function to accept or reject the selection of a data point $d \in X_i$ expressed as follows:

$$f(d_i, l_i, u_i, X_i) = \begin{cases} true, & l_i \leq d_i \leq u_i \wedge min(X_i) \leq l_i < u_i \leq max(X_i) \\ false, & othwerwise \end{cases}$$

Equation 4.1

AND (∧) operator Given a set of bounding values $L = \{l_1, l_2, \dots l_N\}$ and $U = \{u_1, u_2, \dots u_N\}$ with respect to a target variable X_i. Let p be a polyline with a set of data points $p = \{d_1, d_2, \dots, d_N\} \in p$ in N dimensions. We can use Equation 4.1 to determine whether p is in the selection set if all of its containing points D must satisfy the test condition against its target variable $X_i \in X$ as expressed below:

$$\sum_{d_i \in p} \sum_{l_i \in L, u_i \in U} f(d_i, l_i, u_i, X_i) \wedge \dots \wedge f(d_{N-1}, l_i, u_i, X_i), \exists! \, d_i \in X_i, i < N$$

Equation 4.2

In addition, it is equivalent to ignore the AND operator for a variable by relaxing l_i and u_i with the following settings:

$$l_i = min(X_i)$$
$$u_i = max(X_i)$$

Where min and max return the minimal and maximal values so essentially all the data in X_i are selected. Usually, this is the initial state when the dataset is first loaded. The counterpart of AND is the OR operator.

OR (∨) operator The test condition of OR operator is more generous than the AND operator. We consider a polyline is in the selection set if one of its containing point D has satisfied the test condition against its target variable.

$$\sum_{d_i \in p} \sum_{l_i \in L, u_i \in U} f(d_i, l_i, u_i, X_i) \vee \dots \vee f(d_{N-1}, l_i, u_i, X_i), i < N$$

Equation 4.3

Thus, if any of variables in X has set its bounding range to $l_i = min(X_i)$ and $u_i = max(X_i)$ will result in a global data selection.

These operators are useful for restricting or relaxing the selection set but such features come at the cost of intuitiveness. A polyline has no visual continuity so it is very difficult to trace the direction of next line segment at a junction under the circumstance of over-plotting. For example, a polyline appears to be included but may have been filtered out by another AND operator which is far apart. This is illustrated in Figure 4.4 for clarity.

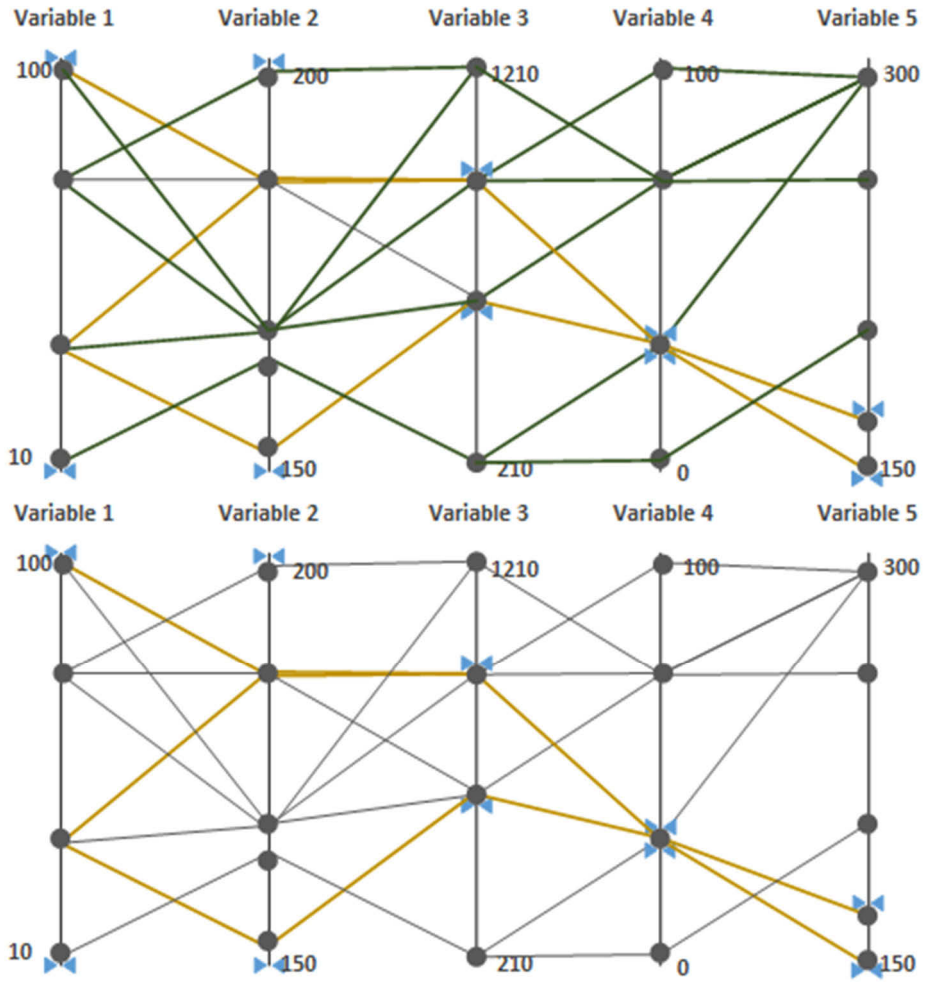


Figure 4.4 The value range model with AND and OR operators in parallel coordinates. The arrows define the upper and lower bound of the value range and the polylines brushed with gold colors indicate the selection. (Top) A demonstration of OR operator, the difference has brushed with green color for easy comparison with the image below. (Bottom) A demonstration of AND operator.

2D AND/OR operator Both l and u need to carry an additional dimension of information denoted as $l_{i,j}$ and $u_{i,j}$ with respect to X_i and X_j. It is trivial to rewrite equations Equation 4.1 and Equation 4.2 for extending 2D operation. Please note that, for 1D data selection, these equations test the condition based on the data value of a given data point rather than its coordinated location on the display.

For visualizing the data selection on a 2D plane, we can apply a rectangular selection technique as we have described previously. Let $l(x, y)$ and $u(x, y)$ be the coordinate for intercepted points of $l_{i,j}$ and $u_{i,j}$ respectively. For each intercepted point, we can draw two $\llcorner 90^{\circ}$ infinite perpendicular lines $[-\infty, \infty]$ past the point. The line must be parallel to the axis such that $l_x \parallel u_x$ and $l_y \parallel u_y$. One can find four intercepted points $(l_x, l_y), (u_x, l_y), (u_x, u_y), (l_x, u_y)$ that consist of a rectangular bound. The AND operator in two dimensional space is only interested in a point $d_{x,y}$ which satisfies the following condition:

$$l_x \leq d_x \leq u_x \wedge l_y \leq d_y \leq u_y$$

Similarly, the condition for the OR operator can be rewritten as:

$$l_x \leq d_x \leq u_x \vee l_y \leq d_y \leq u_y$$

The rectangle enclosure easily reveals the selected data as illustrated in Figure 4.5.

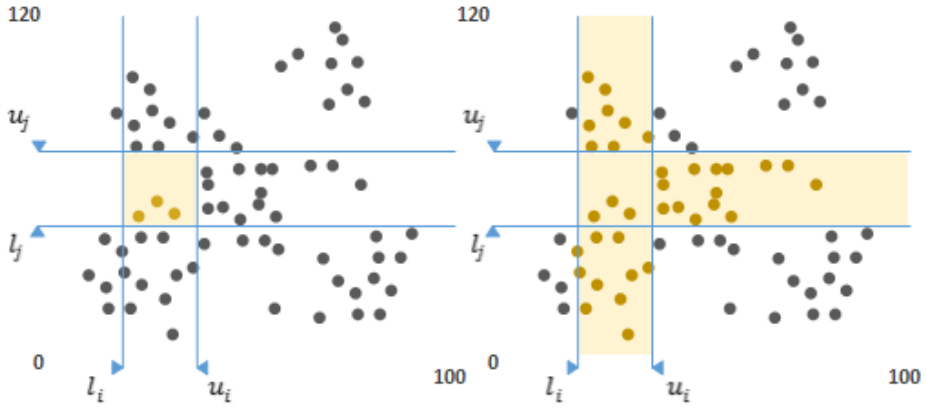


Figure 4.5 An application of the value range model in 2D. The arrows define the upper and lower bound of the value range. (Left) A demonstration of the 2D AND operator. (Right) A demonstration of the 2D OR operator in scatterplot.

This method has a higher degree of accuracy but there are also many shortcomings. The most serious one is the physical cost incurred with the high dimensionality of data if the interaction cost framework studied by Lam [68] is taken into consideration. For example, the physical motions spent will be non-trivial to adjust the value range for 20 or even greater 100 variables. It is extremely time consuming and error prone so for practical

purpose it will deteriorate and eventually become unusable. One can imagine that this kind of situation could occur if one tried to fiddle with more than 100 slider bar widgets.

4.2.3 Point Selection Model

The standard case only needs a mouse click and this is by far the most intuitive behaviour for humans. The minimum drawing unit on the screen is per pixel so the user needs to position the mouse cursor just over the pixels occupied by a geometric primitive for triggering a successful selection. Sometimes, a geometric primitive is misaligned due to antialiasing so the common remedy is to use a tolerance for compensating the misalignment. The basic concept is illustrated in Figure 4.6.

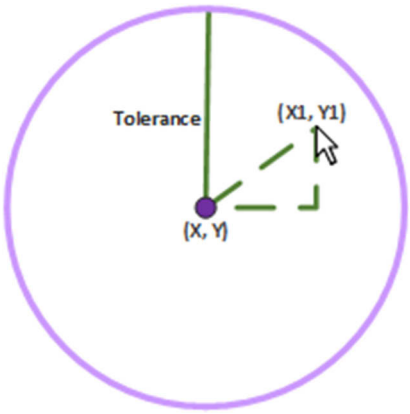


Figure 4.6. Point selection with a tolerance.

Let t, (X, Y) and (X_1, Y_1) denote a tolerance, location of a data point, and location of a mouse cursor respectively. According to the Euclidean distance, if we treat t as a radius then a given data point is considered as selected if the hypotenuse between (X, Y) and (X_1, Y_1) is less than t. Thus, the test function of a point selection can be trivially written as follow:

$$f\big((X,Y),(X_1,Y_1),t\big) = \begin{cases} true & \sqrt{(X_1-X)^2+(Y_1-Y)^2} \le t \\ false, & otherwise \end{cases}$$

The point selection is the most intuitive model but unfortunately, the application in information visualization is generally limited to node-link alike visualizations. The selective accuracy may be fine-grained but it cannot achieve a substantial selection. Figure 4.7 provides an example of a web application with node-link based navigation.

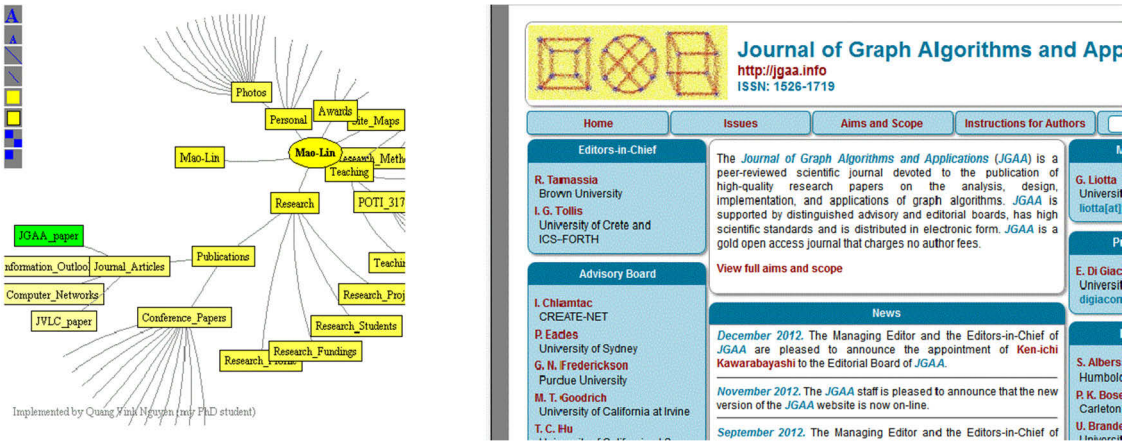


Figure 4.7 A web application with a node-link based navigation. The uUser can select a node on the left view which is a hyperlink and it will reflect to the browser on the right view. This example demonstrates both select and viewing operations.

Currently, there is no application of point selection in parallel coordinates and this is becomes apparent if we scan the relevant literature. However, we have noted from the existing literature that there is a tile-based parallel coordinates contributed by Alsakran et. al. [64] as shown in Figure 4.8. A tile is similar to a node strucutre and the principle of applying a point selection on a tile and a node are alike. Theorically, we see a potential application for point selection on the tile-based parallel coordinates but in practice, it will be chellenging because there is not enough information provided by the author about the association of a tile and the data it represents.

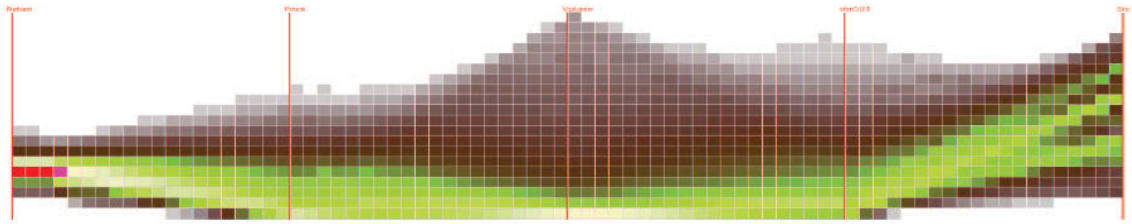


Figure 4.8. A tile-based parallel coordinates. The image is sourced from [64].

There is also an interesting implementation discussed by Shannon et. al. [65] as shown in Figure 4.9 where they partition data into a graph network and expose it by a separate view for the user to interact with them in a point selection fashion. Unfortunately,

the user is only allowed to interact with data indirectly on a separate view and this makes it difficult to understand the selection context.

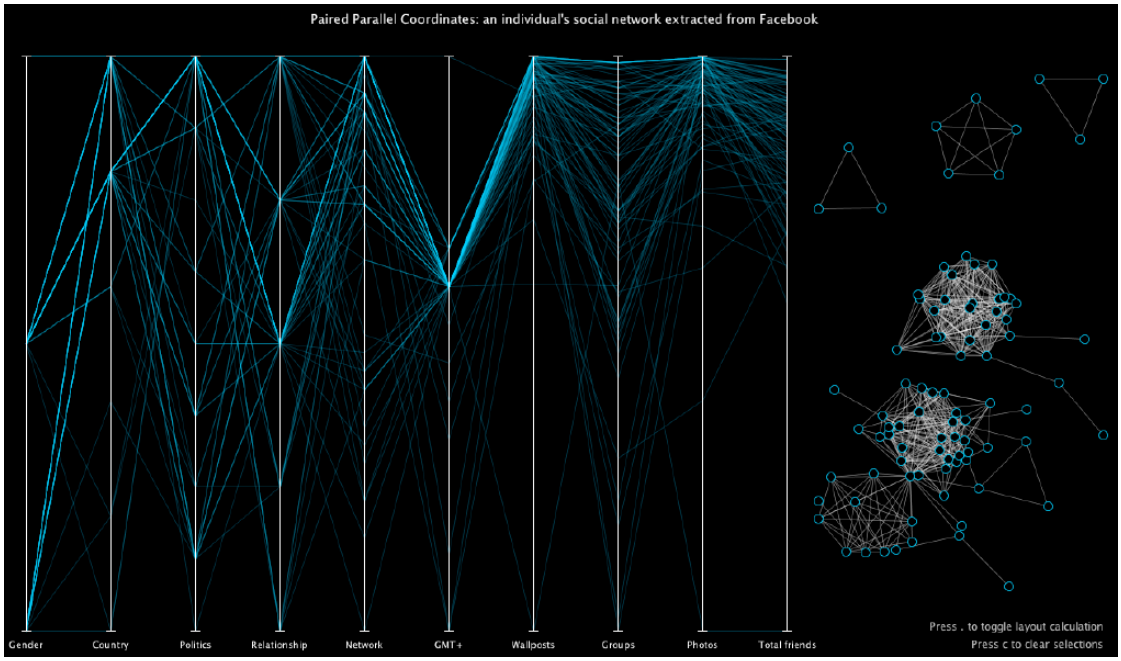


Figure 4.9. An application of the indirect point selection in parallel coordinates. The image is sourced from [65].

4.2.4 Discussion

The models introduced above generally do not provide an excellent user experience in relation to parallel coordinates. In general, overplot and visual clutter are the main barricades of interacting with data in a multidimensional space. The value range model is able to deal with it but it also poses other problems. The following table summarizes the advantages and disadvantages of the models as described previously.

Models	Advantages	Disadvantages
Rectangle Inclusion	Substantial selection.	Poor accuracy and not intuitive.
Value Range	Fine selection granularity.	High interaction cost and not intuitive.
Point Selection	Accuracy, simple and intuitive.	Quantity (Usually, select one data item at a time).

Table 4.2. A summary of the data selection models.

In order to address the issues, we propose a novel technique of *hierarchical virtual node* (HVN) for data selection directly in parallel coordinates with no assumption about the given data. The basic idea behind it is depicted in Figure 4.10. In HVN, data are partitioned hierarchically for variable selection on the basis of location proximity and the display space is filled with virtual nodes for user interaction. Let v_i and d_i be a virtual node and data item respectively where i denotes the hierarchical level and $v_i = \{d_i, d_{i-1}, \dots, d_{i-N}\}$ subjects to $(i - N) > 0$. When a node v_i is mouse clicked, all the data items beneath v_i are included as part of the selection set. The approach offers better flexibility and accuracy with lower physical cost of motion to interact with data in parallel coordinates.

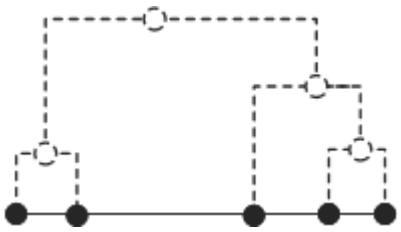


Figure 4.10. Illustration of the hierarchical virtual node design. Space is filled with virtual nodes (circles with broken lines) for user interaction since they are placed hierarchically so it allows the user to perform variable data selection.

The HVN combines the advantages of others with a careful design to minimize the inherited shortcomings. The following table summarizes the features of HVN. The middle and last column describe the operation about how to achieve it in HVN and which model also offers similar features respectively.

Features	How	Similar Model(s)
Substantial selection	Click on the top node (global node) of the hierarchy.	Rectangle inclusion, value range.
Accuracy	Click on the bottom node (data node) or the child node of the hierarchy.	Point selection.

Intuitiveness	Requires only a mouse click on a node.	Point selection.
Lower motion cost	Requires only a mouse click on a node.	Point selection.
Neighboring selection	Data are partitioning into clusters by hierarchical clustering.	Value range (without clustering)
Repeatability of a data selection (*Go back*)	Click on the node again which was clicked previously.	Point selection.
Global and Local Drill Down	Click on any node for interactive data exploration.	N/A
Hierarchical selection	Click on any node and its child nodes are selected.	N/A
Data Density	Observation through the distribution of the virtual nodes.	N/A

Table 4.3. Features of the hierarchical virtual node.

The *repeatability of a data selection* listed in the table is also known as the *go back* feature. In visual analytics, one frequently needs to reproduce the previous view again for gaining recollection. To achieve this, one has to redo the previous steps that were carried out. In the rectangular selection model, it is difficult to exactly redraw the same rectangle again but such a task is trivial in the HVN.

There are several advantages of the HVN over other techniques. First, there is no need to specify a value range. Quantization is always difficult which usually requires the user to understand the data characteristics. Second, it provides the granularity of multi-level selection of data with greater flexibility to quickly explore the patterns between the nearest neighbors. Third, virtual nodes provide the visual distribution of the data. The distribution of the nodes also suggests the distribution of the continuous variable. Finally, it has a minimal interaction cost because the entire operation can be activated by a single mouse click rather than interpreting a sequence of mouse operations. Overall, the compliexity of the HVN is greater than others and the technical details will be discussed in the following section.

4.3 Implementing the HVN

This section is dedicated to the technical discussion of the HVN. The basic idea of the HVN was born in a discussion with Dr. Mao Lin Huang. The key points are briefly extracted below to the best of our remembrance where Q and A mean *the question asked by me* and *the answer provided by Dr. Mao Lin Huang* respectively.

> *Q: ... existing techniques claiming to be fully interactive did not really mention how they deal with the situation where the display is overplotted and cluttered ..., ... also poly based geometric primitives are very difficult to interact with ....*
>
> *A: ... clutter is caused by the polylines..., why don't we use a node to represent the data for direct data selection? ...*
>
> *Q: ... point selection is good, but it is not efficient to select data in the multidimensional data space..., there are too many data to select one by one ...*
>
> *A: ... why not partition the data hierarchically?...*
>
> *Q: ... OK, the idea is feasible but how do we use a node to represent the polyline? ...*
>
> *A: ... how about we use a virtual node? ...*

The above interaction serves to reveal the originality of the HVN.

4.3.1 System Overview

An interactive parallel coordinates system with the tight integration of HVN has been successfully implemented. A simplified version of the system flowchart is outlined in Figure 4.11. The simplification is in a form where some minor steps have been skipped but they will not affect the overall integrity of the flow chart. In the figure, a diagram is also attached to a step for visually explaining the state of the corresponding process. Overall, the whole implementation of the HVN can be condensed into Figure 4.11. In the following sections each section will be dedicated to each process in the figure.

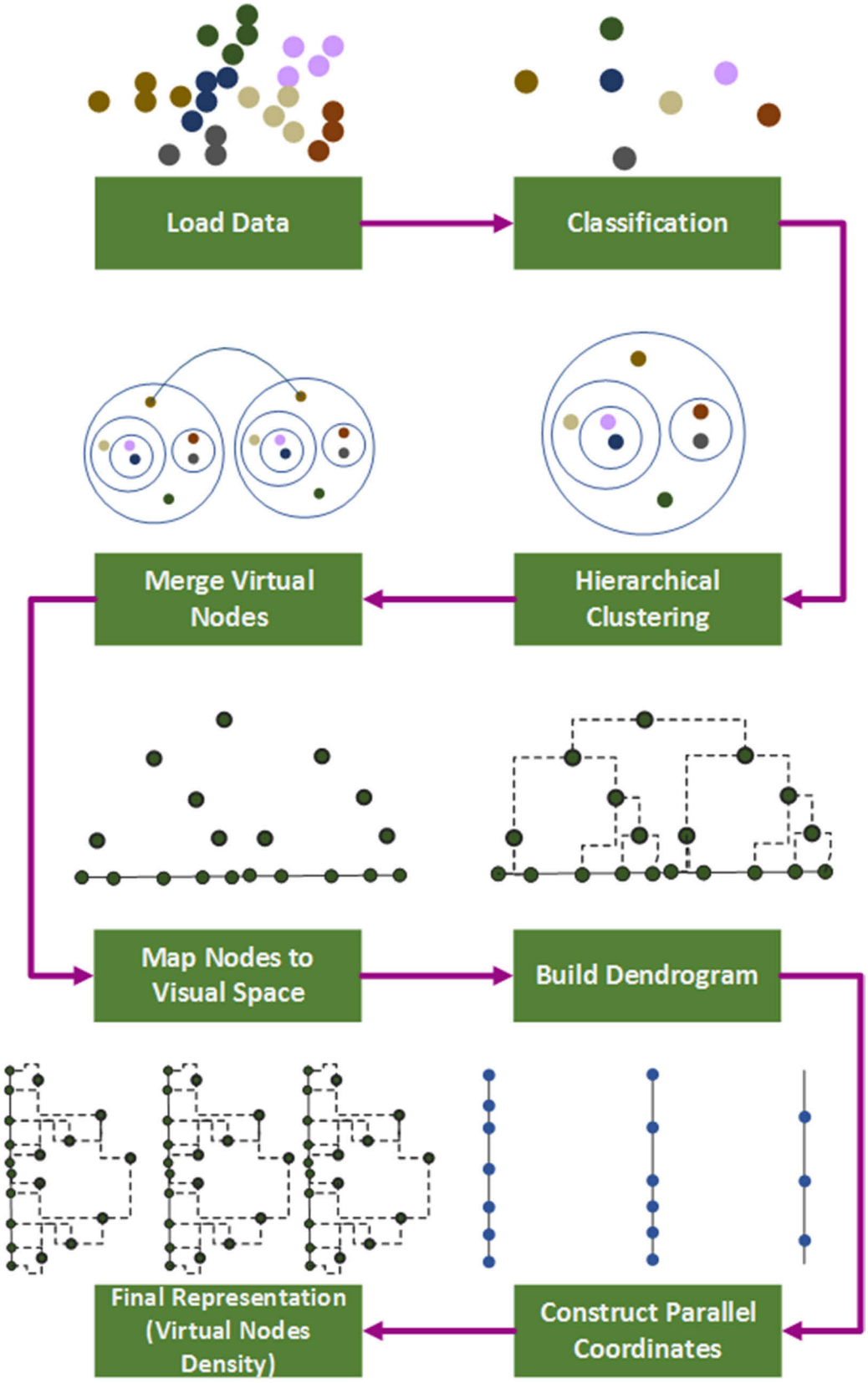


Figure 4.11. System flowchart of the HVN based parallel coordinates.

4.3.2 Data Classification

The goal of applying classification is to remove data redundancy. This step enhances the computational efficiency for the next stage of hierarchical clustering. Given a variable X_i which contains a set of data points $D = \{d_1, d_2, \dots, d_M\}, \forall d \in X_i$. Let $EQ: A \rightarrow B$ be a function that defines the equivalence relation. Any two arbitrary elements d_i and d_j in X_i are said to be equivalent if and only if they meet the condition $EQ(d_i, d_j) = EQ(d_j, d_i)$. The expression of EQ can be written as:

$$EQ(d_i, d_j) = \begin{cases} 1 & d_i = d_j \Leftrightarrow d_i \cap d_j \neq \emptyset \\ 0, & otherwise \end{cases}$$

Equation 4.4

Given the equivalence relation defined in Equation 4.4, we can obtain the equivalence classes denoted as $D/IND(X_i) = \{E_1, E_2, \dots, E_N\}$ by partitioning D into a disjoint subset using the following indiscernibility relation written as:

$$IND(X_i) = \{(d_i, d_j) \in D: EQ(d_i, d_j) = 1\}$$

Equation 4.5

These equivalence classes will be passed to the next stage of the system pipeline. In summary, the purpose of this step is to ensure data uniqueness and they are only logically removed from the dataset in order to avoid redundant computation.

4.3.3 Non-parametric Partitioning by Hierarchical Clustering

The core task of data partitioning involves a clustering technique. Michaud [69] provided a great work that covers the well-known methods. As the name suggests, the HVN uses hierarchical clustering. Although, K-means [43] is another alternative, it requires an input parameter of K clusters to be known in advance which is not desirable in our case.

Hierarchical clustering [44] is a non-parametric technique and probably one of the most widely used methods in data mining. Given a set of items, the method partitions them into a set of disjoint clusters hierarchically in the following form.

$$G = \{g_1 \cup g_2, \dots \cup g_N\}^1, \dots, \{\{g_1 \cup g_2\} \cup \{\dots\}, \dots \cup \{g_N\}\}^\delta : g_i \cap g_j = \emptyset, i \neq j$$

Where δ denotes the depth of the hierarchy. The hierarchy of the clusters are built iteratively by merging two clusters with greatest similarity to form a new cluster in each iteration. The similarity is determined by the choice of an agglomerative algorithm which will be described latter.

The first step in hierarchical clustering transforms the data items in each variable into a distance matrix. Univariately, this is straightforward and one simply computes the Euclidean distance on a pairwise data items as:

$$dist(d_i, d_j) = |d_i - d_j| \; where \; d_i \neq d_d$$

Equation 4.6

The result of $dist(d_i, d_j)$ is carried forward in the corresponding $[i, j]$ element within a matrix as:

$$D = \begin{pmatrix} dist(d_1, d_1) & \cdots & dist(d_1, d_N) \\ \vdots & \ddots & \vdots \\ dist(d_N, d_1) & \cdots & dist(d_N, d_N) \end{pmatrix}$$

Where D denotes the distance matrix rather than data value matrix. The diagonal elements $dist(d_\mathrm{i}, d_\mathrm{i})$ are always 0. Hierarchical clustering is nonparametric with no assumption about the target K cluster known in advance. Instead, it requires a stopping rule to break the process prematurely when the optimal number of clusters have been determined within the range $1 \leq k \leq |\{D\}|$. Fortunately, we build an entire hierarchy so finding a suitable stopping rule is not a concern in this step. The strategy of hierarchical clustering applied here is agglomerative which builds up the hierarchy from the bottom-up.

Initially, hierarchical clustering initializes each data point in D into a singleton cluster that is $f: \{D\} \rightarrow \{G\}$ and the cardinality is one such that $\forall_{g \in G} |\{g\}| = 1$. Recall that, D has already been preprocessed by the classification mentioned in Section 4.3.2. In the next iteration, it recursively merges a pairwise cluster to form a new cluster $G_{i,j} = \{G_i \cup G_j\}$ based on an objective function. The inter-cluster distance $\overrightarrow{G_{i,j}G}$ is updated by the linkage criterion. At the end of each iteration, G is subtracted by 1 such that $G = G - 1$ and the process continues until $|\{G\}| = 1$ which represents a global set situated at the top of the hierarchy.

In many sciences, the homogeneous requirement is commonly quantized by the distance measure. Thus, the objective function can be simplified to be a minimization problem. Alternatively, one might be interested in maximizing the measure but it really depends on the problem domain. Give a set of clusters G, the aim is to find a nearest pairwise g_i and g_j with the minimum path by the following equation.

$$\sum_{g \in G} argmin \left| g_i - g_j \right| : \left| g_i - g_j \right| = \sqrt{(g_i - g_j)(g_i - g_j)}$$

Equation 4.7

Obviously, the objective function in Equation 4.7 is essentially a Euclidean distance which measures the length between g_i and g_j in one direction. This is the most desirable distance measure in our case since we expect the merge of clusters to fully respect to their location proximity in the following order.

$$\{d_1, d_2, d_3, d_4\} \Rightarrow \{\{d_1, d_2\}, d_3, d_4\} \Rightarrow \begin{cases} \{\{d_1, d_2\}, \{d_3, d_4\}\} \\ \{\{d_1, d_2, d_3\}, d_4\} \end{cases} \Rightarrow \{\{d_1, d_2, d_3, d_4\}\}$$

In other words, we want neighboring clusters to be merged progressively without jumping since we cannot reorder[6] the data in the screen space. In visualization, the distance of $\overrightarrow{XY}$ in the screen space typically implies their relative distance in data space $|X - Y|$ proportionally.

There are three well-known linkage criterions to update the inter-cluster distance for the newly formed cluster $G_{i,j}$. They are single link, complete link and average link and these concepts have been well explained by Day and Edelsbrunner [70]. In short, single linkage searches the shortest inter-cluster distance between g_i and g_j from the adjacency matrix and Equation 4.6 can perfectly be reused in this case. Complete linkage finds the maximum inter-cluster distance and is given by:

$$\sum_{g \in G} argmax \left| g_i - g_j \right|$$

6 The concept of reordering only applies to variables rather than data items since doing so cannot guarantee a data item will respect its origin in the coordinate system.

In other words, single link and complete links aim to minimize and maximize the inter-cluster separation respectively whereas, average linkage is a hybrid approach between the two. The conceptual explanation of cluster separation is illustrated in Figure 4.12.

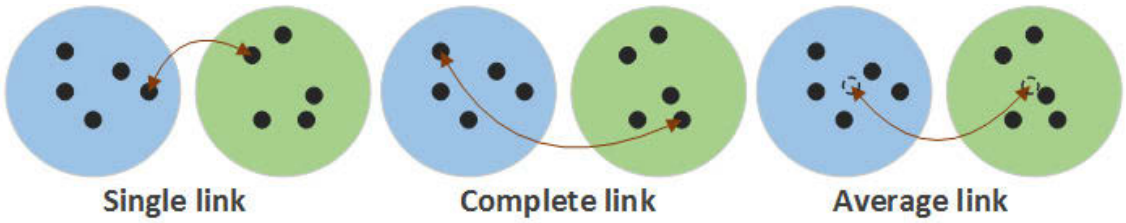


Figure 4.12 An illustration of the single, complete and average linkages. The diagram shows how the distances between clusters are computed.

In updating the inter-cluster distance, we have applied the average linkage. The average linkage algorithm is also known as Unweighted Pair Group Mean Arithmetic (UPGMA). As the name suggests, it computes the distance of a new cluster $G_{i,j}$ from the average values of pairwise G_i and G_j. The equation is defined as follows:

$$UPGMA(G_i, G_j) = \frac{1}{|G_i| \times |G_j|} \sum_{d_i \in G_i} \sum_{d_j \in G_j} dist(d_i, d_j), i \neq j$$

Equation 4.8

Given a set of variables $X = \{X_1, X_2, \ldots, X_N\}$, Algorithm 4.2 provides a versatile function for constructing the hierarchical clustering.

1. **procedure** $HierarchicalClustering(X = \{X_1, \ldots, X_N\})$
2. $V = \emptyset$
3. **for each** X_i **in** X
4. $G = \sum_{d_i \in D} \sum_{d_j \in D} dist(d_i, d_j) : i \neq j$
5. **while** $|\{G\}| > 0$ **do**
6. $p_{i,j} = \sum_{g \in G} argmin \left| g_i - g_j \right| : i \neq j$ /* Find the clusters to merge. */
7. $G_{i,j} = \left\{ G_{p_i} + G_{p_j} \right\}$ /* Merged two clusters. */

8. $G = G - G_i$ /* Remove cluster. */
9. $G = G - G_j$ /* Remove cluster. */
10. **for** $i \leftarrow 0$ **to** $|\{G\}|$
11. $G_i = UPGMA(G_i, G_{i,j})$
12. **end for**
13. $G \leftarrow G + G_{i,j}$ /* Add the merged cluster. */
14. **end while**
15. $V = G$
16. **end for**
17. **return** V
18. **end procedure**

Algorithm 4.2 An implementation of the hierarchical clustering using average linkage.

Where $p_{i,j}$ denotes a temporary variable which holds the indices for G_i and G_j. Given the example of a multidimensional data, our approach of hierarchical clustering is univariate and this is also evident in Algorithm 4.2.

In our system pipeline, this process partitions data into logical groups hierarchically as illustrated in Figure 4.13. The data hierarchy remains logical and the task of the next process will map the virtual nodes into the visual space for visualization.

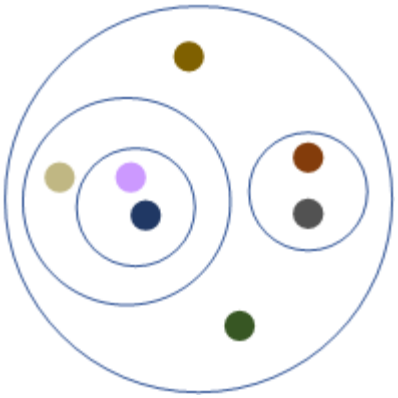


Figure 4.13. Logical groups partitioned by the hierarchical clustering.

4.3.4 Mapping Virtual Nodes into Visual Space

Let V be an output produced from the previous process which holds the hierarchical clustering result for the whole variable set X with the following properties:

$$V = \{v_1, v_2, \dots, v_N\} : \forall v = \{g_1, g_2, \dots, g_N\}, v \to G \to D \; w.r.t \; X_i$$

It is important to classify here for avoiding confusion with the notations used. We have used D, G and V to imply data points, data groups and vertices in Section 4.3.2 for data classification, Section 4.3.3 for hierarchical clustering and for visual space mapping respectively. It is just a convention adopted and they ultimately refer to the same raw data. Technically, v simply describes a logical grouping G of input data D with respect to a given variable X_i and it remains abstract at this stage. Hence, we need to allocate the physical region for v in the visual space before they can do any useful work.

Cluster level ℓ provides important information for positioning a virtual node correctly in the hierarchy and it was skipped in the previous section just to simplify the notation of G. Given a v, it is critical to explore its deepest level ℓ otherwise, its screen coordinate will be misplaced. Considering the example depicted in Figure 4.14, if we only explore the right branch of the topmost node then ℓ is 1 which is clearly wrong and its coordinate will be mapped to a lower location than it is supposed to be.

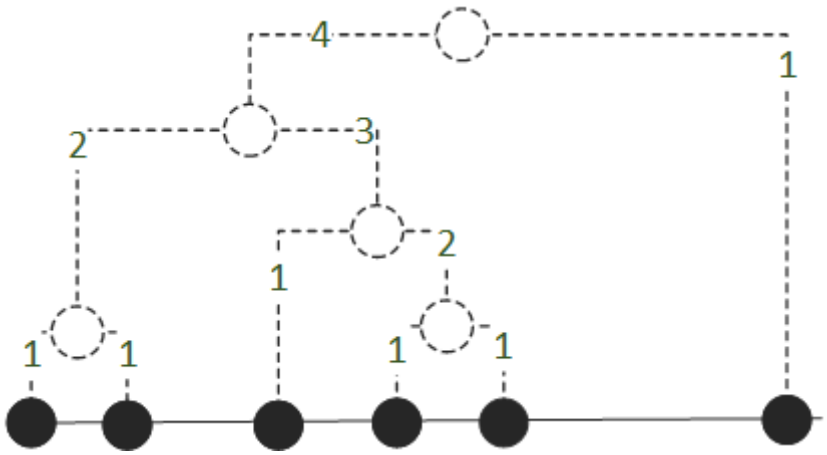


Figure 4.14. Virtual node depth in the hierarchy. Nodes are annotated with levels.

One can determine the deepest cluster level ℓ for any given node v by traversing its children recursively. Since our implementation is recursive, the precaution has been taken

to persist with the local maximum found at each branch. The technique is also known as depth first search [71] as implemented in Algorithm 4.3.

```
1.    procedure DFS(v, currentDepth)
2.       depth = currentDepth /* Keep track the local maximum. */
3.       while v.HasMoreChildren( ) do
4.          depth = depth + 1
5.          for j ← 0 to v.ChildrenCount then
6.             d = currentDepth /* Save children's state. */
7.             depth = max(DFS(v.ChildAt(j), d), depth)
8.          end for
9.       end while
10       return depth
11.   end procedure
```

Algorithm 4.3 An algorithm that computes the depth of a virtual node in the hierarchy.

Each cluster has been assigned a value called the *centroid* denoted as τ which is a value which represents the center of a cluster and does not necessarily serve as part of a data member of that cluster. A centroid is always a middle point between its left and right children. Recall that, centroid is a data value so it can be perfectly mapped to the y coordinate with respect to a target variable X_i for a given virtual node v. Let β be a constant representing the interval (aka width) between levels, the x coordinate can be obtained by:

$$X_v = X_{X_i} + (\beta \times \ell_v)$$

Where X_{X_i} denotes the X coordinate of a vertical axis for X_i in parallel coordinates (the implementation that was already provided in Algorithm 2.1) and ℓ_v denotes the level (or depth) with respect to a given node. Accordingly, the Y coordinate is obtained by:

$$Y_v = PointToScreen(X_i, \tau_v)$$

Where $PointToScreen$ is a wrapper function of Equation 2.1 as provided earlier and τ_v denotes a centroid of v which was worked out from Equation 4.8. The algorithms are completely provided in Algorithm 4.4. Virtual nodes are always laid out from the top node so the usage is $LayoutVirtualNode(V_1, X_i)$ where V_1 and X_i are the top node and the target variable respectively.

---

1. **procedure** $LayoutVirtualNodes(v, X_i)$
2. $Orientation = None$
3. **if** $(i + 1) < |X|$ **then**
4. $\quad Orientation \coloneqq Right$
5. **else**
6. $\quad Orientation \coloneqq Left$
7. **end if**
8. $width = 0$
9. **if** $Orientation$ **is** $Left$ **then**
10. $\quad width = \left(X_{X_{(i-1)}} - X_{X_i}\right)/2.0$
11. **else**
12. $\quad width = \left(X_{X_i} - X_{X_{(i+1)}}\right)/2.0$
13. **end if**
14. $\ell_v = DFS(v, 0)$
15. $\beta = width/(\ell_v + 1)$ /* Work out the width of the interval */
16. *LayoutVirtualNodes*$(v, X_i, interval, Orientation)$
17. **end procedure**
18.
19. **procedure** *LayoutVirtualNodesRecursive*$(v, X_i, \beta, Orientation)$
20. $offset = 0$
21. **if** $Orientation$ **is** $Right$ **then**
22. $\quad offset = (\beta \times \ell_v)$
23. **else**
24. $\quad offset = -(\beta \times \ell_v)$

---

25. **end if**
26. $Y_v = PointToScreen(X_i, \tau_v)$ /* Work out the y coordinate here */
27. $X_v = X_{X_i} + offset$ /* Work out the x coordinate here */
28. **for** $j \leftarrow 0$ **to** $v.ChildrenCount$ **then**
29. *LayoutVirtualNodesRecursive* $(v.ChildAt(j), \beta, Orientation)$
30. **end for**
31. **end procedure**

Algorithm 4.4. Algorithms of mapping a virtual node to the screen coordinate.

The width defined in the algorithm is a maximal height and it is half way to the adjacent variable. Please refer to Figure 4.15 for clarity.

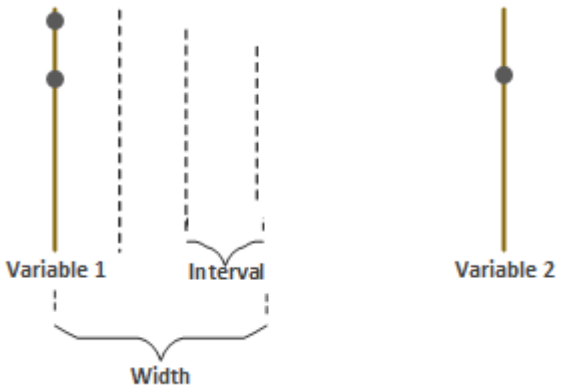


Figure 4.15. Virtual node layout definitions.

In addition, one should also note that the *orientation* is taken into consideration while transforming a virtual node into the screen coordinate. This is because the orientation for positioning virtual nodes for the last vertical axis shall face toward left rather than right as depicted in Figure 4.16.

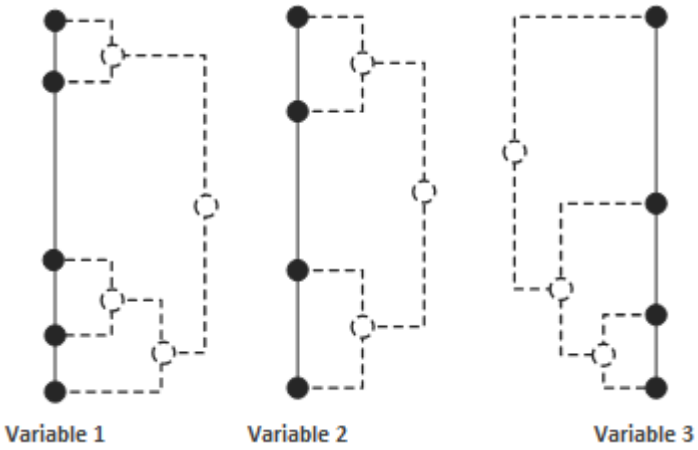


Figure 4.16. Illustration of the virtual node layout.

At this stage, the basic skeleton of the HVN has emerged. In the next section, a dendrogram will be constructed. Actually, the interactive part of the HVN is fully functioning without dendrogram but the sole purpose is to visualize a hierarchy for visual association of a virtual node with its parent and child nodes.

4.3.5 Building a Dendrogram

A *dendrogram* is a tree-like structure which is used to visualize the hierarchy of the clustering arrangement. The skeleton of a dendrogram can be materialized by connecting all the virtual nodes and the purpose is to provide a visual association between parent and child nodes. Let $\overrightarrow{l_1 l_2}$ denote a connection from a child to its parent node, there are two lines l_1 and l_2 required as shown in Figure 4.17.

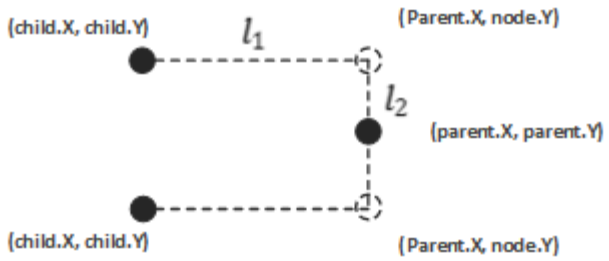


Figure 4.17 Connection of virtual nodes.

Given v_{child} and v_{parent}, the start ρ_1 and end point ρ_2 of l_1 can be derived as follow:

$$X_{\rho_1} = X_{v_{child}}$$
$$Y_{\rho_1} = Y_{v_{child}}$$
$$X_{\rho_2} = X_{v_{parent}}$$
$$Y_{\rho_2} = X_{v_{child}}$$

Similarly, the start ρ_1 and end point ρ_2 of l_2 can be written as follow:

$$X_{\rho_1} = X_{v_{parent}}$$
$$Y_{\rho_1} = Y_{v_{child}}$$
$$X_{\rho_2} = X_{v_{parent}}$$
$$Y_{\rho_2} = X_{v_{parent}}$$

A complete implementation of drawing a dendrogram is provided in Algorithm 4.5 and again, we start the drawing from a topmost node.

```
1.    procedure DrawDendrogram(v)
2.      while v.HasMoreChildren( ) do
3.        for j ← 0 to v.ChildrenCount then
4.          v_child = v.ChildAt(j)
5.          DrawLine(X_{v_child}, Y_{v_child}, X_v, Y_{v_child})
6.          DrawLine(X_{v_parent}, Y_{v_child}, X_{v_parent}, Y_{v_parent})
7.          DrawDendrogram(v)
8.        end for
9.      end while
10.   end procedure
```

Algorithm 4.5. An implementation of drawing a dendrogram.

There are three types of nodes namely, global, data and virtual nodes. Figure 4.18 provides a diagram describing their locations. The lowest cluster level is called a *data node* since it is a singleton cluster containing only a data point. The *global node* sits at

the root level that holds the hierarchical topology entirely. The rest are called the *virtual nodes* meaning that they are interpolated for interaction. Strictly speaking, a global node is implicitly a virtual node.

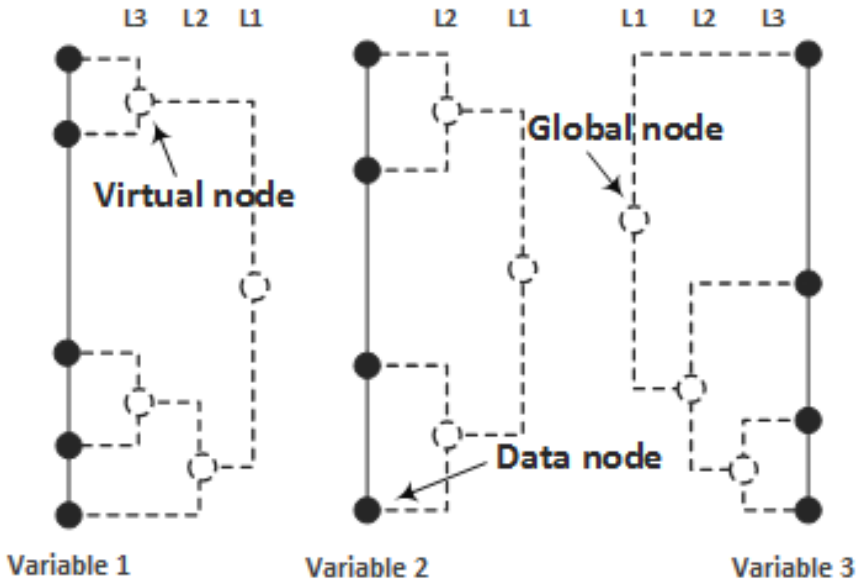


Figure 4.18 Types of virtual node.

4.3.6 Constructing Parallel Coordinates

In this section, we will be discussing the geometric drawings and the integration of HVN into parallel coordinates. There are three geometric primitives currently provided in our parallel coordinates system namely, polyline, Bezier curve and Bezier virtual nodes.

4.3.6.1 Polyline

Polyline is a classic primitive adopted by Inselberg [13] as well as many parallel coordinates systems. A complete implementation of classic parallel coordinates has already been provided in Algorithm 2.1 so we will not repeat it here.

A polyline is a simple way to model the path of a multidimensional data but it is often criticized in terms of its geometric discontinuity where user can only observe partial paths if they overlap. This problem is visualized in Figure 4.19.

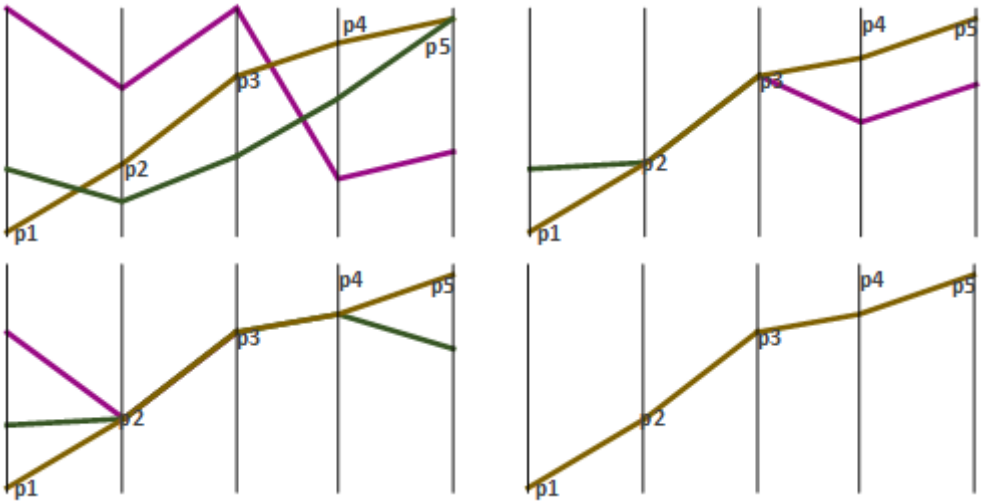


Figure 4.19. Severities of the overlapped polylines. (Top Left) No overlapping, (Top Right) Unable to observe $\overrightarrow{p_1p_2p_3}$ of the pink polyline. (Bottom Left) Unable to observe $\overrightarrow{p_2p_3p_4p_5}$ of the pink polyline. (Bottom Right) Unable to observe others except the brown polyline.

The implementation of polyline primitive is conceptually trivial and a snapshot of our system implementation is provided in Figure 4.20

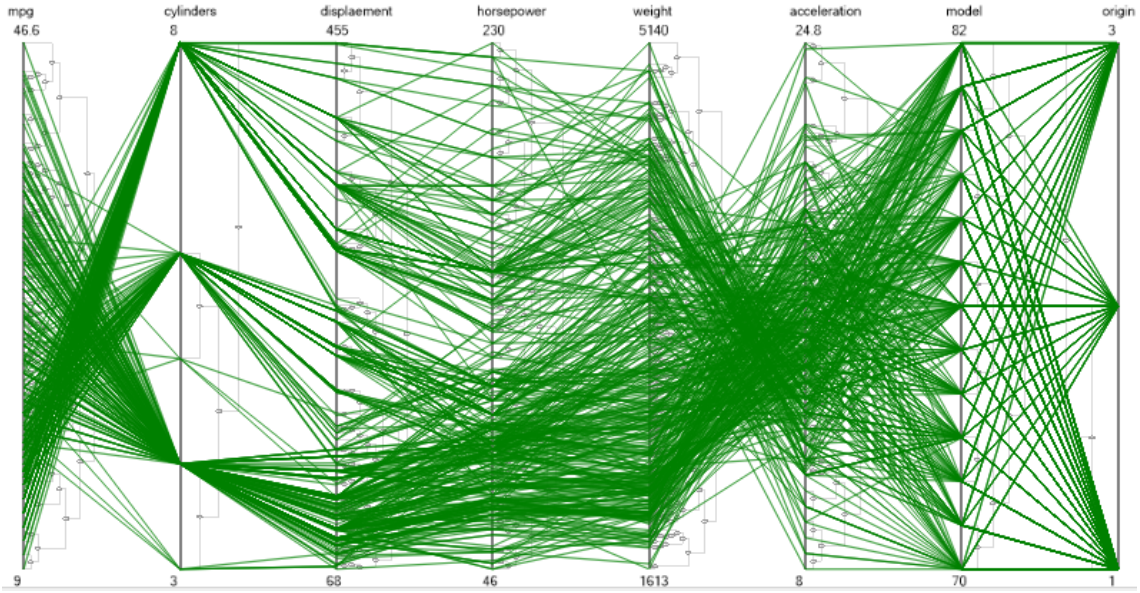


Figure 4.20. A snapshot of polyline primitive in our system.

4.3.6.2 Bezier Curve

A Bezier curve [72] is a parametric curve with the mathematical basis based on the Bernstein polynomial [73]. Bezier curve is known as parametric in the sense that the path is controlled by a set of control points $C = \{P_0, P_1, \dots, P_{N-1}\}$ and a parameter t. Graham [74] has discussed an application of curve in parallel coordinates and the advantage is the dynamic path of oscillation which can be controlled by adjusting control points. By contrast, the path of a classic polyline is always static and there is no way to change it to avoid the overplot.

The properties of a Bezier curve are listed as follows:

- P_0 and P_{N-1} are the end points and lie on a curve.
- Middle control points from P_1 to P_{N-2} do not either pass through or lie on the curve as shown in Figure 4.21. However, the linear Bezier curve is an exception.
- $\overrightarrow{P_0P_1}$ is a tangent line $\perp$ at the point P_0. Similarily, $\overrightarrow{P_2P_3}$ is a tangent line $\perp$ at the point P_3.
- Invariance with affine transformation.

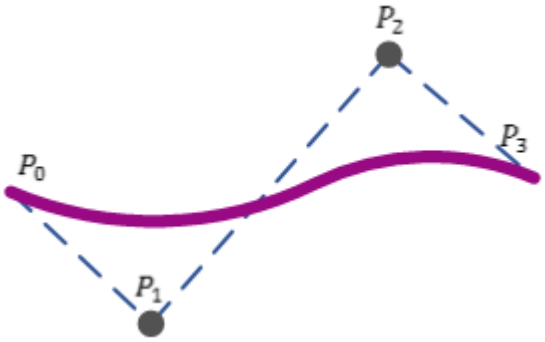


Figure 4.21. Bezier curve with control points.

Mathematically, a Bezier curve of degree n is generally written as:

$$P(t) = \sum_{i=0}^{n-1} P_i \times B_i^n(t)$$

Equation 4.9

Where t specifies a ratio along a line such that $t \in [0,1]$ and B_i^n is a Bernstein polynomial expressed as:

$$B_i^n(t) = C_i^n(1-t)^{n-1}t^i$$

Where C_i^n is a binomial coefficient given as:

$$C_i^n = \frac{n!}{i!\,(n-1)}$$

These theorems have been exhaustively discussed in the existing literature and widely accessible but we suggest, in particular, an excellent note written by Sederberg [75]. Equation 4.9 is a generalized expression and for a Bezier curve with a degree of 3, we can expand the polynomial to the following form:

$$B(t) = (1-t)^3P_0 + 3(1-t)^2tP_0 + 3(1-t)t^2P_1 + t^3P_1, \forall P_i \in \{P_o, P_1, \dots, P_{N-1}\}$$

Equation 4.10

Recall that the nonlinear problem is often approximated linearly. The output of $B(t)$ evaluates to a point which lies on a Bezier curve, as shown in Figure 4.22 for $t \in [0,1]$. In general, a smaller step of t generates a more smooth curve and vice versa. However, from a practical point of view, t shall be chosen as just enough for smoothing a Bezier curve otherwise it will impact the memory usage and performance of the system. In our implementation, t increments at an interval of 0.05 so $B(t)$ is actually invoked 21 times to approximate just one Bezier curve segment. In Figure 4.24 provides an experiment with various interval values where 0.5 and 0.1 have produced a C^{-1} continuous curve since one can easily observe the discontinuities.

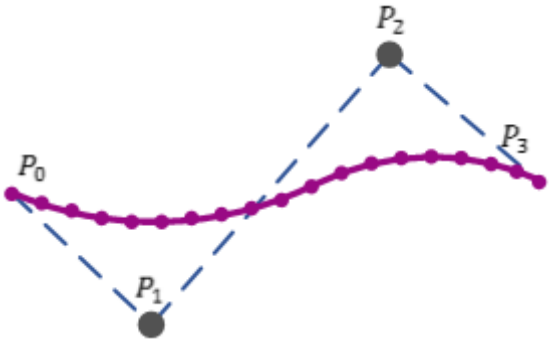


Figure 4.22. Evaluation of points in a Bezier curve.

In our approach, we construct a Bezier curve segment with 4 points where the start and end points are data points with respect to their target variables. For example, given a

multidimensional data row $P = \{d_1, d_2, \dots, d_M\}$, there are $M - 1$ Bezier curves constructed as $b_1 = \overrightarrow{d_1 d_2}, b_2 = \overrightarrow{d_2 d_3}, \dots, b_{M-1} = \overrightarrow{d_{M-1} d_M}$. These Bezier curves are connected together to form a single C^0 continuous curve, as illustrated in Figure 4.23. Alternative, there is another approach called B-spline curve, but it is more complex and will not be implemented here.

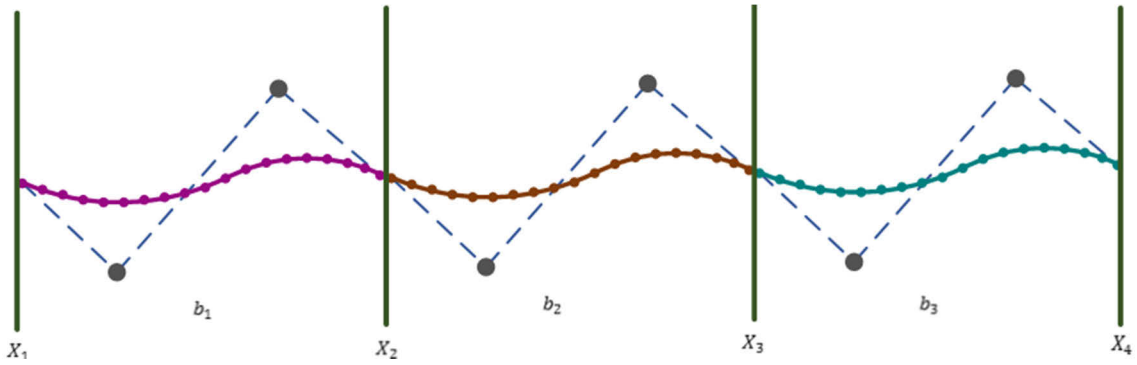


Figure 4.23. Bezier curve segments.

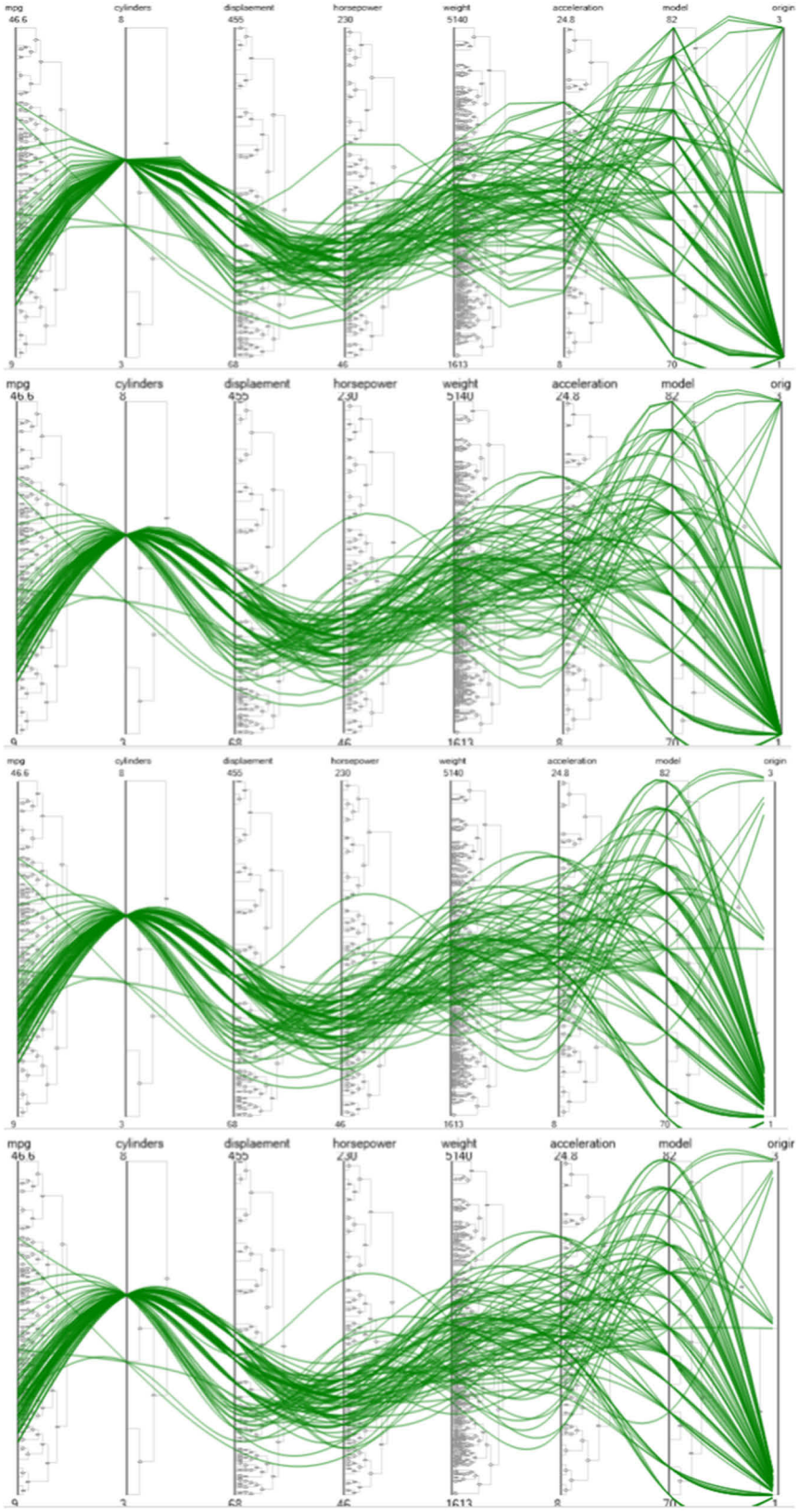


Figure 4.24. Bezier curves with various intervals of t. The value of t increments at an interval of 0.5, 0.2, 0.1 and 0.05 from the top to the bottom image.

4.3.6.3 Bezier Virtual Nodes

Bezier virtual nodes is not a new geometric primitive. It is considered to be an extension of the Bezier curve by drawing non-uniform curve segments that must pass through virtual nodes to form a single curve.

Definition 4.1 *Knots are end points of a curve, the start and the end point. We say a curve is uniform if the knots are equally spaced and vice versa.*

The goal of the Bezier virtual nodes aims to prevent the overplot of the virtual nodes. One feasible way is to interpolate them as knots in a curve. Figure 4.25 illustrates the idea.

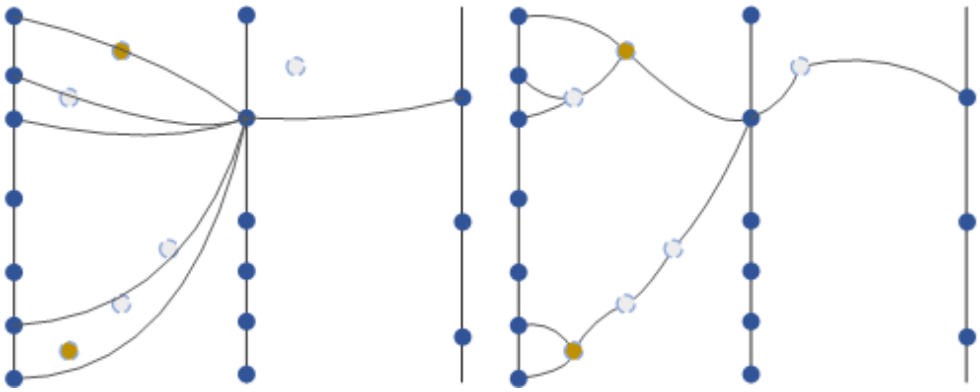


Figure 4.25 Geometric drawing of the selected data. (Left) A geometric drawing with overplot of the virtual nodes. (Right) A geometric drawing by treating virtual nodes as end points.

The geometric basis is based on the previous section. Considering virtual nodes as knots, there are more Bezier curve segments generated by inserting knots and a Bezier virtual nodes curve is formed by connecting them all together, as illustrated in Figure 4.26.

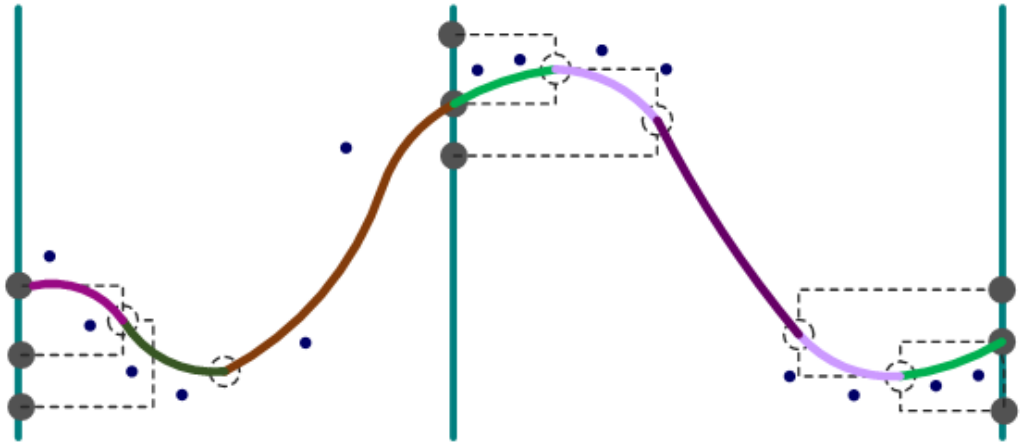


Figure 4.26. Bezier virtual nodes style drawing.

The challenge of this drawing comes from the mapping for a mixture of virtual and data nodes to form a complete curve. Please refer to Figure 4.18 again for the definition of the node types. For example, suppose there are 10 data rows selected by a mouse click over a virtual node before the system can draw them, then it needs to work out a complete path that it must pass through (which consist of N virtual and data nodes just for a data row) and such task is far from trivial.

This challenge is not an issue in the polyline and Bezier curve because their paths are formed without any consideration of the virtual nodes. In the following, we will describe how the system responds to a virtual node click and also how we have optimized our data structures to achieve high system performance.

Tracing nodes in virtual space In our initial implementation, there were two separate data structures maintained, one for storing the data values in a data matrix D and another one for virtual nodes which persisted in a tree structure denoted as T. Given a virtual node at location v_i, the system had to cross reference D and T multiple times. The time consuming nature of this implementation was approximately $O(n^3)$ which is computationally expensive if a given dataset is considered large $N \to \infty$. As a consequence, the system performance was poor. A better optimization is given in the following.

Optimization by tracing the virtual node in data space One feasible performance improvement can be made by mapping a virtual node and its hierarchical level to a

corresponding data point and virtual dimension respectively. In other words, the additional data of $\ell \times M$ dimensions are added to the dataset with respect to X_i where the size of ℓ is equal to the number of levels. Thus, the data matrix D needs to be redefined with respect to each variable X_i as:

$$D = \{D_1, \dots, D_M\} \cup [\ell \times M]$$

The greatest advantage of this implementation is the ability to treat virtual nodes and real data points consistently but it comes at the cost of space complexity. However, we are mainly concerned with reduction of the time complexity to achieve a reasonable response time for better user experience.

The data table in Figure 4.27 illustrates the conceptual implementation which contains one data dimension and three visual dimensions shaded in black and blue colors respectively. Recall that, a virtual dimension is inserted to represent each level in the hierarchical clustering. The NaN means a null node and does not represent any end point in a geometry. For example, there are two null nodes in the fourth row of the data table so the algorithm will draw only one line segment. Note that, if a NaN appears in a normal dataset it is called missing data. Dealing with missing data is outside the scope of this thesis hence they will be removed in the preprocessing step for simplicity.

Real Data	Virtual Data			Real Data	Virtual Data	
Dimension 1	P1	P2	P3	Dimension 2	P1	P2
20	NaN	18.25	NaN	20	20	18.25
17	16.5	18.25	NaN	8	21	18.25
16	16.5	18.25	NaN	43	16.5	18.25
10	NaN	NaN	7	24	NaN	NaN
6	NaN	NaN	7	19	NaN	NaN
3	2	4	7	21	15	14
1	2	4	7	1	2	4

Figure 4.27 Redefinition of a data matrix. The data matrix is redefined to support the storage of the virtual node and virtual dimension.

Suppose, if D1 (P2 with a value of 18.25) and D2 (P1 with a value of 2) in Figure 4.28 are clicked then the top three rows and bottom two rows are selected. Thus, we only need to translate the data points to the screen coordinates since the virtual nodes have

already been stored in the matrix D. One can see that such optimization greatly simplifies the overall problem by completely removing the cross references between separate data structures.

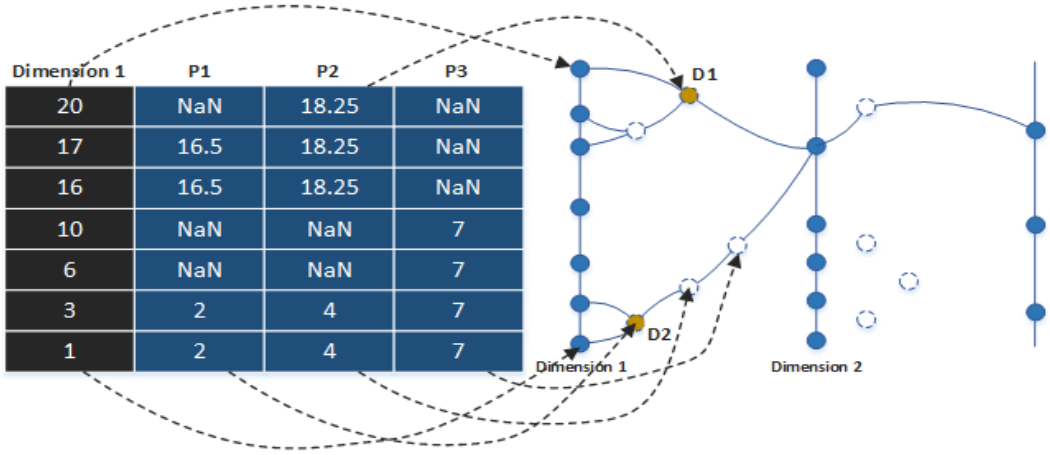


Figure 4.28 An illustration of geometric mapping of data and virtual node to parallel coordinates.

Following the procedures described above, we can obtain the result as illustrated in a snapshot from our system in Figure 4.29.

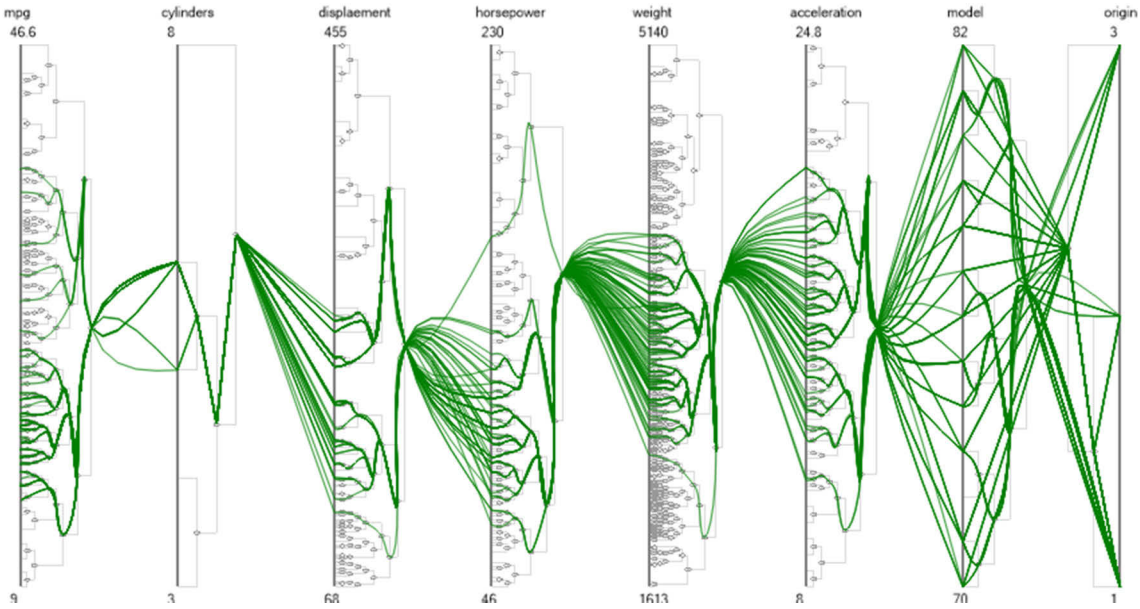


Figure 4.29. A snapshot of the Bezier virtue nodes drawing in our system.

Interestingly, by comparing these 3 drawings with a dataset which contains 58776 (12×4898) data points, we were surprised that the Bezier virtual node incurs less visual clutter due to its hierarchical arrangement of the curves and the result is also shown in Figure 4.30 where the geometric primitives are polyline, Bezier curve and Bezier virtual nodes for the top, middle and bottom image respectively. It is obvious that the rendering by the Bezier virtual nodes can reduce the overplot of the visualization.

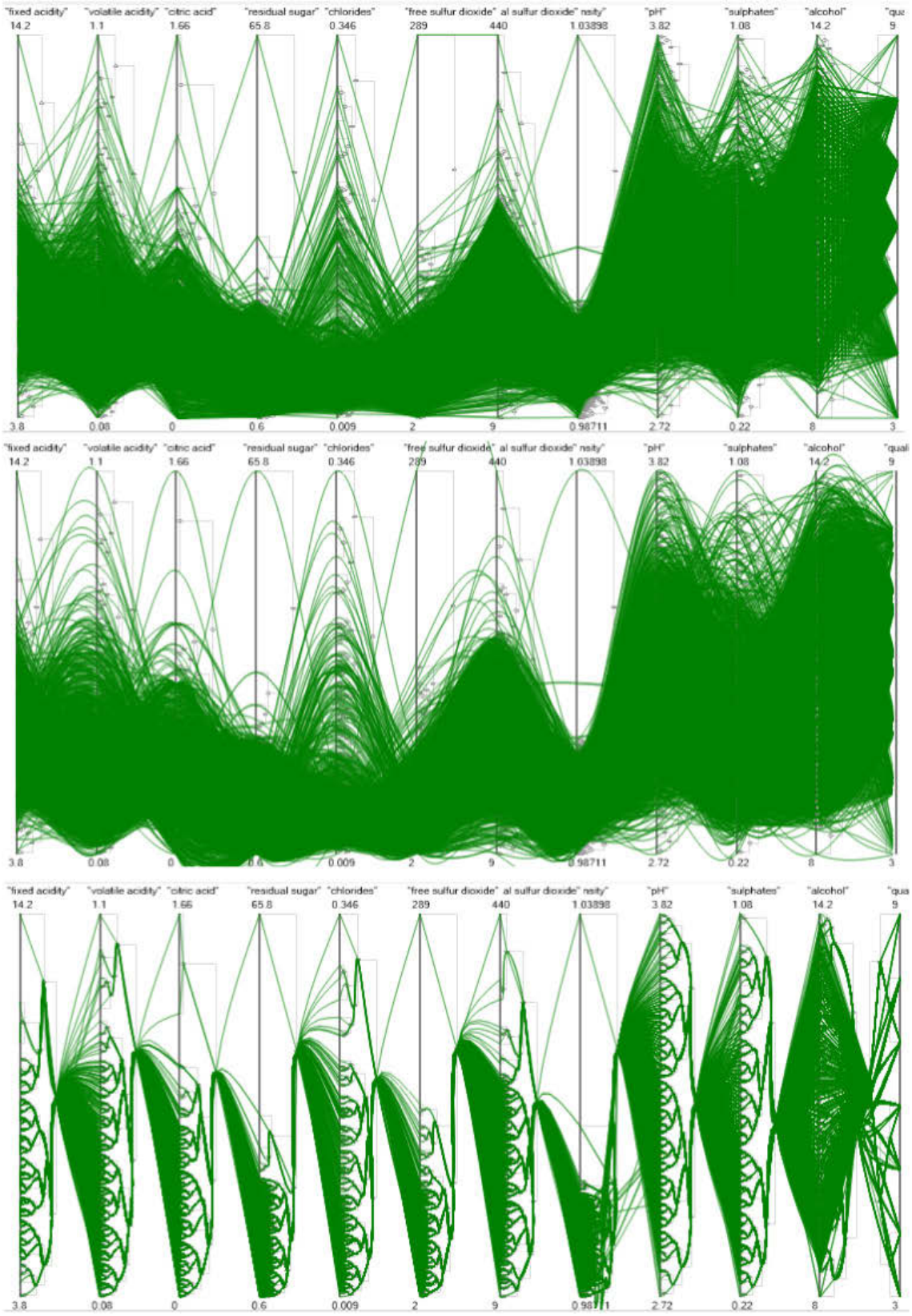


Figure 4.30. Comparison of the overplot severity between geometric primitives.

4.3.7 Overview Presentation by Virtual Nodes Density

All the technical implementations of the HVV have been fully described above and a real snapshot of the system is illustrated in Figure 4.31. By default, our overview hides all the geometric objects except the virtual nodes. Thanks to the virtual nodes, we are able to modify the Visual Information Seeking mantra proposed by Shneiderman [56] from *overview first* to *overview first by virtual nodes*. This allows the user to perceive the distribution of the data density through the distribution of the virtual nodes in an elegant way.

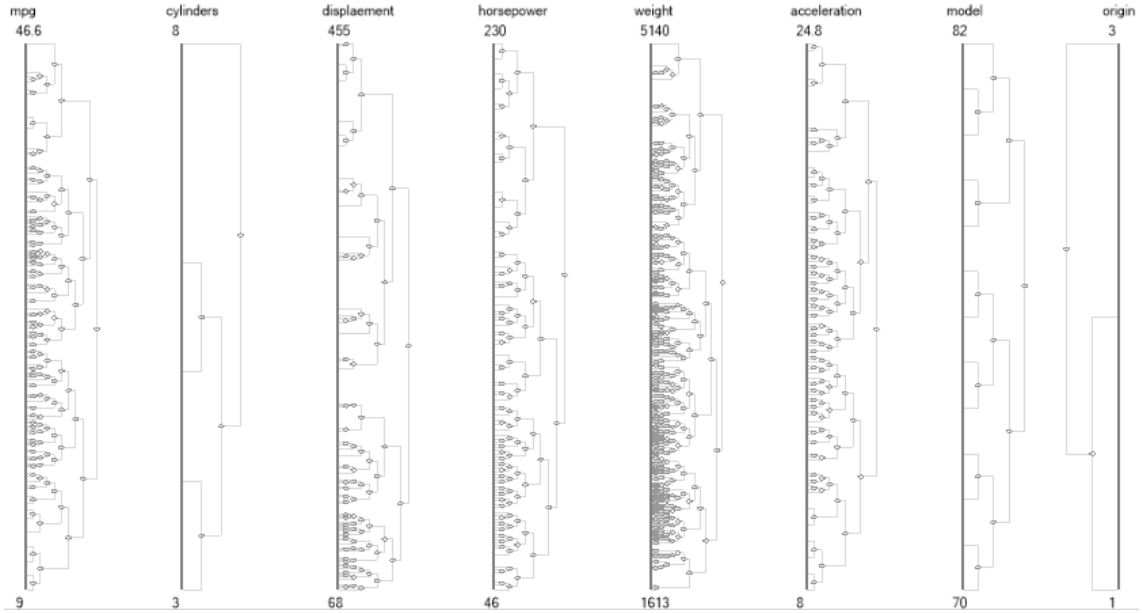


Figure 4.31. Overview presentation of the HVN in parallel coordinates.

Alternatively, the observation of the density distribution can be achieved by embedding histogram to parallel coordinates and it was actually our initial consideration. However, we have soon realized that the virtual nodes provide a better way of perceiving the data density in reality. Nonetheless, an analysis of the histogram will be provided below.

Histogram [76] is the simplest nonparametric method for estimation of the data distribution of a random variable. Figure 4.32 provides an example of the histogram for visualizing data distribution.

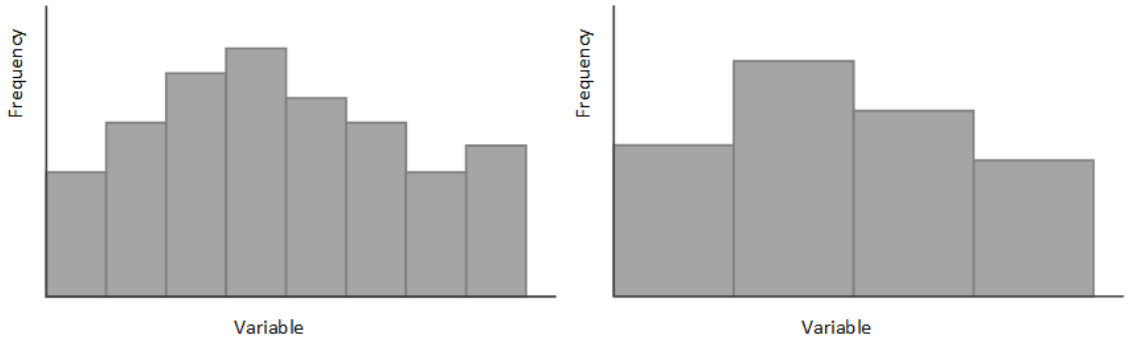


Figure 4.32. An illustration of ordinary histograms. The bin size and width are the most crucial factors that determine the shape of the histogram. For example, the shape of the data distribution has changed from 8 (left) to 4 bins (right). The discontinuities are the artifact of the chosen bin size.

The procedure of the histogram begins by classifying data univariately into bins and then counting the occurrence of each disjoint bin $B = \{b_1 \cup b_2 \cup \dots b_N\} : b_i \cap b_j = \emptyset, i \neq j$. The following equation is used to find the number of bins.

$$k = \frac{(X_{max} - X_{min})}{h}$$

Equation 4.11

Where h denotes the bin width. The choice of proper width h is difficult since the variance is not unity. To address this problem, one can alternatively work out the optimal k first by Sturges's [77] formula as follows:

$$k = \frac{(X_{max} - X_{min})}{log_2 N + 1}$$

Equation 4.12

By substituting k obtained in Equation 4.12 back to Equation 4.11 to work out the bin width h. It defines the cut point for counting the data occurrence fall within a bin range expressed as follows:

$$\big(X_{min} + (h \times i)\big) < d \leq \Big(X_{min} + \big(h \times (i+1)\big)\Big)$$

Equation 4.13

Where i denotes the bin index and X_{min} is the minimum value of a target variable X_j. If the range condition specified in Equation 4.13 has met for a given data point d then we simply increment the cardinality as $n_i \leftarrow |b_i| + 1$ where n_i holds the number of samples in bin b_i. The histogram density function for estimating a data point is therefore given as:

$$\hat{f}(d) = \frac{n_i}{nh}$$

Where the width of b_i must straddle the data point d and it can be easily determined by using Equation 4.13 and n and h denotes the total samples and bin width respectively. The probability of a data point which falls within the width of a bin is given as

$$p_i = \int_{b_i} f(d)\, dx$$

The parallel coordinates system implemented by Hauser el al. [32] embeds the histogram as shown in Figure 4.33. However, such as approach significantly increases the visual loading of overall visualization by displaying a mixture of geometric objects and histograms simultaneously. In contrast, our *overview first by virtual nodes* approach presents an uninterrupted way of perceiving the data density.

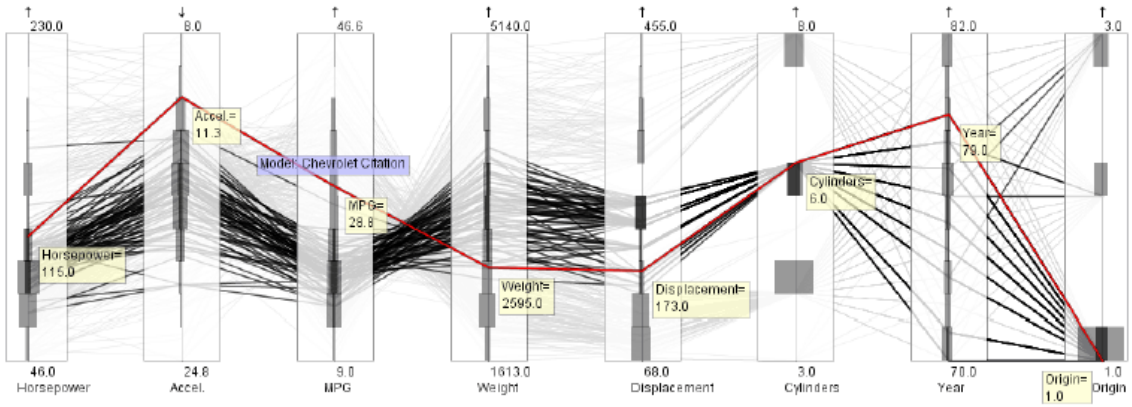


Figure 4.33. A parallel coordinates with histograms embedded. The image is collected from [32].

4.4 Performance

A system has been successfully implemented which was written in C# and OpenGL Shader Language (GLSL) in order to gain GPU hardware acceleration. The Open Tookkit library (OpenTK) [78] is used for OpenGL binding by exposing C functions to C#. The technology is also called *Interop* in .NET. The performance benchmarking is based on the hardware platform depicted in Figure 4.34.

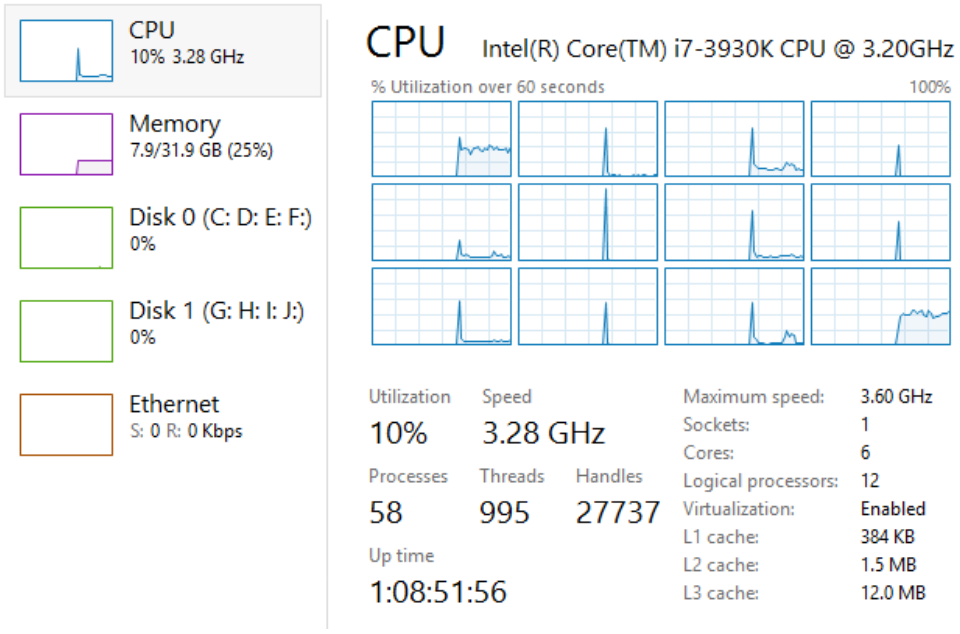


Figure 4.34. Hardware environment for benchmarking.

There are 4 datasets used in benchmarking and the performance metric has been provided in Table 4.4. The *frame per second* (FPS) is an important counter since it affects the responsiveness of the visualization when responding to a user request. In most cases, the FPS can satisfy the request except the NYTS dataset where all the geometric primitives are set to visible and then the FPS will suddenly drop to 9. Note that, *FPS (Geometric primitives invisible)* means visualization is in an initial state as shown in Figure 4.31.

Dataset	Size ($M \times N$)	FPS (Geometric primitives invisible)	FPS (Geometric primitives visible)
Car	8 × 392	477	334
Wage	11 × 534	730	476
Wine	12 × 4898	113	65
NYTS	116 × 14776	81	9

Table 4.4. Performance measurements of the system. Car and Wage datasets were obtained from http://lib.stat.cmu.edu/datasets/, the source of the Wine dataset is https://archive.ics.uci.edu/ml/datasets/Wine and NYTS dataset was obtained from http://www.cdc.gov/tobacco/data_statistics/surveys/nyts/.

4.5 Discussion

In summary, this chapter presents a novel technique to interact with data in parallel coordinates. We described several data selection models but they generally do not work well in multidimensional space. The basic idea behind HVN is to interpolate virtual nodes directly in parallel coordinates for hierarchical data selection. A node structure is an intuitive interface of interaction because it has a duality of data and coordinates. Overall, the HVN is a revolutionary way of data selection which provides a truly direct interaction with data and is also effective under the circumstances of overplotting and visual clutter. It is also a core technique in our interactive framework in which all the interactive tasks are built on the basis of it. To recall the advantages and features of the HVN, please refer to Table 4.2 and Table 4.3.

Chapter 5 Interactive Techniques for Visual Analytics

In this chapter, we will focus on the HVN-oriented interactions in visual analytics. With the advent of the visual analytics, we have noted the Visual Analytics mantra described by Keim [11] as:

Analyze first, show the important, zoom, filter and analyze further, details on demand.

Obviously, it is based on the Visual Information Seeking mantra proposed by Shneiderman [56]. Keim [79] which also describes the process of visual analytics as:

The visual analytics process is a combination of automatic and visual analysis methods with a tight coupling through human interaction in order to gain knowledge from data.

This mantra reinforces the importance of iterative analysis and interaction in connection with attempts to explore a dataset because a fully automatic data analysis can only be accomplished if the problems are well-defined and this is usually not possible in most real world cases. Therefore, the user remains the final decision maker to drive the direction of the whole process through the iterative interactions with a visualization. Recall that we have merged Yi et al.'s [14] seven layers model into three so the following sections will focus on our three layers model.

5.1 Task by Dynamic Selection

Interactive data selection is an indispensable component of an effective visualization. It improves the usability by accurately translating the intention of mouse operations into a selection query for data manipulation. Many parallel coordinates visualizations claiming

to be interactive did not really focus on the *Select* interaction; layer 1, defined by J. S. Yi el al. [14]. Overall, there are so far none of the existing techniques that could well achieve the functions covered by the *Select* layer of J. S. Yi's [14] seven-layer interaction model or the *Highlighting (or Selection)* layer defined. Perhaps, it is too difficult to have mouse click (or selection) on the data item which is virtually represented by a polyline (or curve) in strongly overlapped parallel coordinates.

The HVN is the core technique of data selection in our model. It helps users to perform subsequent analytics tasks in an efficient and practical manner. In the following sections, we will describe some tasks in the dynamic selection layer based on the HVN.

5.1.1 Interact with Data by the HVN

Materializing a point selection model in parallel coordinates is one of contributions that the HVN has made. The use case of HVN simply requires a mouse click on a target node of interest. Recall that again, virtual nodes storage has been optimized in Section 4.3.6.3 for tracing a complete end-to-end path of a virtual node efficiently. Given a random mouse click captured at the location (X_{mouse}, Y_{mouse}). We know that a virtual node is strictly placed on an interval boundary β such that $\beta = width/(\ell_v + 1)$ and this equation was already defined in Algorithm 4.4. With this information, we can apply the function below for querying which variable and level ℓ to search for in the data matrix D.

$$f(X_i, X_{mouse}) = \begin{cases} i & \sum_{i=0} X_{X_i} + (i \times \beta_{X_i}) = X_{mouse} : X_{X_i}(i \times \beta_{X_i}) < X_{X_{i+1}} \\ -1 & otherwise \end{cases}$$

Equation 5.1

Where i denotes a relative offset with respect to its base X_i and -1 indicates an invalid index because the negative value cannot be used to index a matrix column. Suppose X_i, X_{X_i}, X_{mouse} are Dimension1, 200 and 210 respectively and let β_{X_i} be 5. The above function returns $\ell = 2$ which translates to an offset starting from X_i for looking up D. The next step involves building a range query by transforming Y_{mouse} to a data value denotes as d_{mouse}. The value transformation can be easily achieved by rewriting Equation 2.1 as:

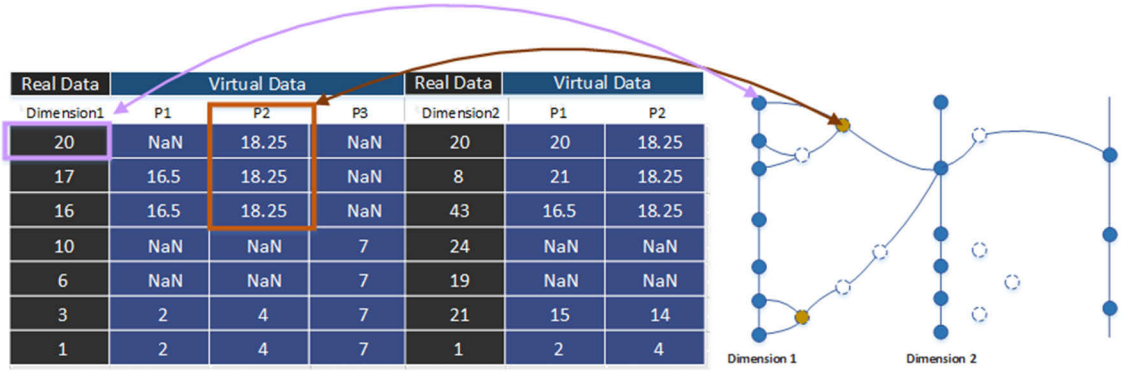
$$d_{mouse} = \left(\frac{\left(Y_{mouse} - Y_{X_i}\right) \times \left(max_{X_i} - min_{X_i}\right)}{height_{X_i}} \right) + min_{X_i}$$

Equation 5.2

Suppose Y_{mouse}, Y_{X_i}, max_{X_i}, min_{X_i} and $height_{X_i}$ are 90.8, 0, 20, 1 and 100 respectively. The above equation returns ≈ 18.25. Now, we are able to query the data matrix D with the following pseudo query expressed as:

SELECT ROWS FROM Dimension1 WHERE $\ell = 2$ *AND* $d = 18.25$

Of course, the query above is used for illustration purpose and is not an actual case. We shall obtain 3 data rows as illustrated in Figure 5.1. Please note that $\ell = 0$ returns a real data value rather than a virtual data (which represents a virtual node).



Real Data	Virtual Data			Real Data	Virtual Data	
Dimension1	P1	P2	P3	Dimension2	P1	P2
20	NaN	18.25	NaN	20	20	18.25
17	16.5	18.25	NaN	8	21	18.25
16	16.5	18.25	NaN	43	16.5	18.25
10	NaN	NaN	7	24	NaN	NaN
6	NaN	NaN	7	19	NaN	NaN
3	2	4	7	21	15	14
1	2	4	7	1	2	4

Figure 5.1. Data query in the HVN. It shows the mapping of the real data and virtual data to the parallel coordinates.

The implementation for the procedure of querying selected data is provided in Algorithm 5.1.

1. **procedure** $HitTest(X_{mouse}, Y_{mouse})$
2. $\Upsilon = \emptyset$ /* A list of selected row indices, initialize to empty. */
3. **for each** X_i **in** X
4. $\ell = HitTestX(X_i, X_{mouse})$

```
5.        if ℓ ≥ 0 then
6.            d_mouse = HitTestY(X_i, Y_mouse)
7.            for i = 0 to |D|
8.                if D[i, ℓ] = d_mouse then
9.                    ϒ ← i
10.           end for
11.       end if
12.   end for
13.   return ⟨X_i, ϒ, ℓ_{X_i}, ℓ⟩
14. end procedure
```

Algorithm 5.1. An algorithm for hit test.

Where D, $[*]$, $HitTestX$ and $HitTestY$ denote a global data matrix, indexing operation and the wrapper functions for Equation 5.1 and Equation 5.2 respectively. The output is a tuple $\langle X_i, \Upsilon, \ell_{X_i}, \ell \rangle$ which holds a target variable, selected row indices, base column index for X_i, and relative offset from its base ℓ_{X_i} for indexing a virtual node. Please refer the notations in Figure 5.2 for clarity.

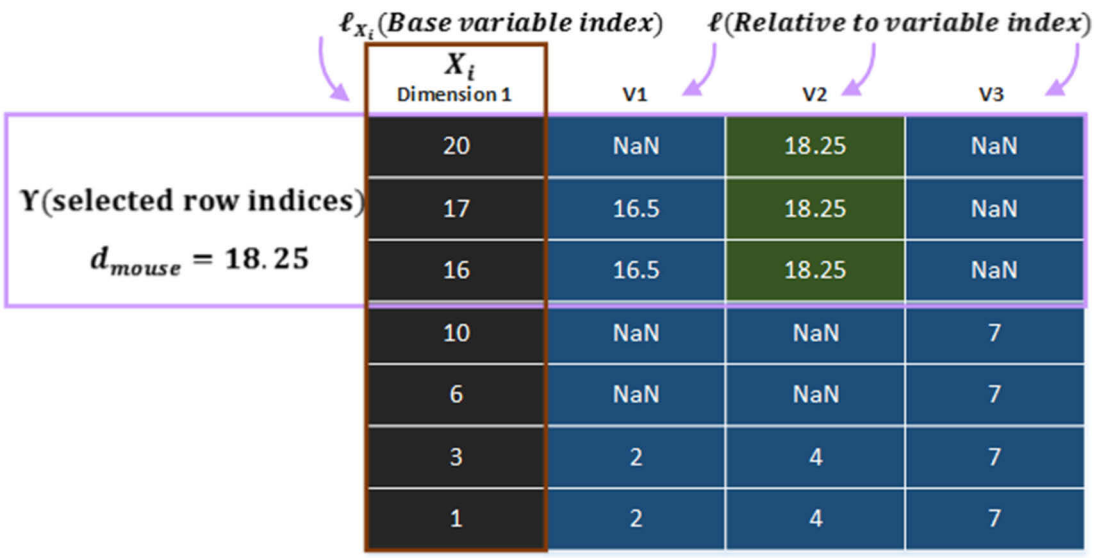


Figure 5.2. Notations used for query the global data matrix.

Where ℓ_{X_i} is a static value associated with X_i and ℓ is obtained by translating the y-coordinate of a mouse click to an index relative to ℓ_{X_i} in the runtime. The tuple answers enough technical questions below for us to look up D.

1. What is the target variable X_i?
2. How many row are selected as well as their indices?
3. If a selected node is a data node such that $\ell = 0$, we use ℓ_{X_i}, otherwise it is a relative offset from ℓ_{X_i} such that $\ell_{X_i} + \ell$ (see Figure 5.1).

Since we are equipped with all the information provided by the tuple, the next step is to look up the data values and transform them into the screen coordinates connected by a geometric primitive. In Figure 5.3, we have demonstrated the operation of direct data selection by clicking on a virtual node as indicated in the top image and the middle and bottom diagrams show the selected data in polyline and Bezier virtual nodes styles. Note that, we only need to transform virtual data into screen coordinates if the specified geometric primitive are Bezier virtual nodes otherwise we skip them.

In general, the HVN completely changes the way that user interacts with data directly in parallel coordinates which is more simple, intuitive and accurate than any existing techniques.

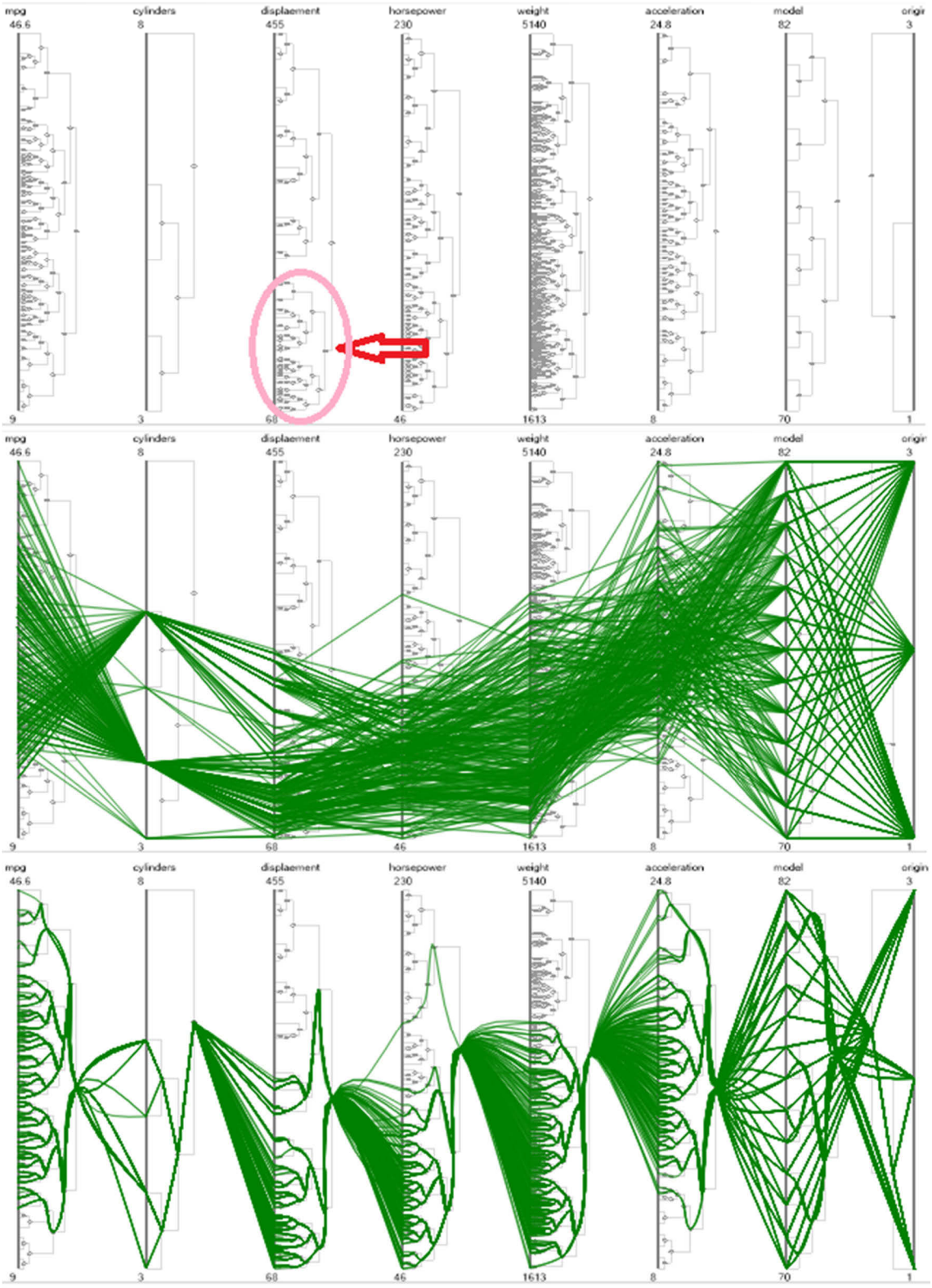


Figure 5.3. Direct data selection via a virtual node.

5.1.2 Dynamic Brushing via HVN

Brushing is commonly used to discern multidimensional data patterns by visual differentiation. In 1994, Ward [36] first proposed a concept of *N dimensional brushing* that can be used to highlight N dimensional data items which fall within a user-specified subspace (or sub-region) in either scatterplots or parallel-coordinates. By using brushing, a subset of data items (polylines) within specified value ranges of one or more dimensions can be highlighted (or focused) for viewing the structure of data patterns. This allows users to gain insights into the spatial relationships of N dimension. Lately, several alternative brushing methods have been proposed in the parallel coordinates visualization. For example, Hauser et al. [32] in 2002 presented a concept of angular brushing as an extension of Ward's standard brushing to facilitate data subsets grouping and highlighting by a technique called angular constraint. Both techniques integrated the composite brushing and focus+context technique to further improve the visual exploration in parallel coordinates. In 2003, Yang et al. [45] contributed an automatic and manual brushing mechanism to the parallel coordinate geometry called Visual Hierarchical Dimensionality Reduction (VHDR).

Nowadays, brushing has become an integral component in parallel coordinates. Our system equipped with the HVN has provided an excelling interface for users to carry out such tasks with the simple steps as follows:

1. Observe the data density through the distribution of the virtual nodes,
2. Apply the color,
3. Click on a target node of interest, and
4. Go to step 1 if the task is not yet finished.

One can see that the contribution of the HVN greatly enhanced the interactivity and usability of the parallel coordinates. In other models, this can be cumbersome. For example, one gets to first figure out the maximal and minimal values of a data group for value range filtering in value range model before brushing.

Figure 5.4 shows the outcome of brushing from the procedures described above which only took approximately ≈16 seconds to brush 5 data groups and the time was mostly spent on choosing the next color. Of course, the timing can be considerably reduced by random coloring. That is, a distinctive color is generated after every brushing operation.

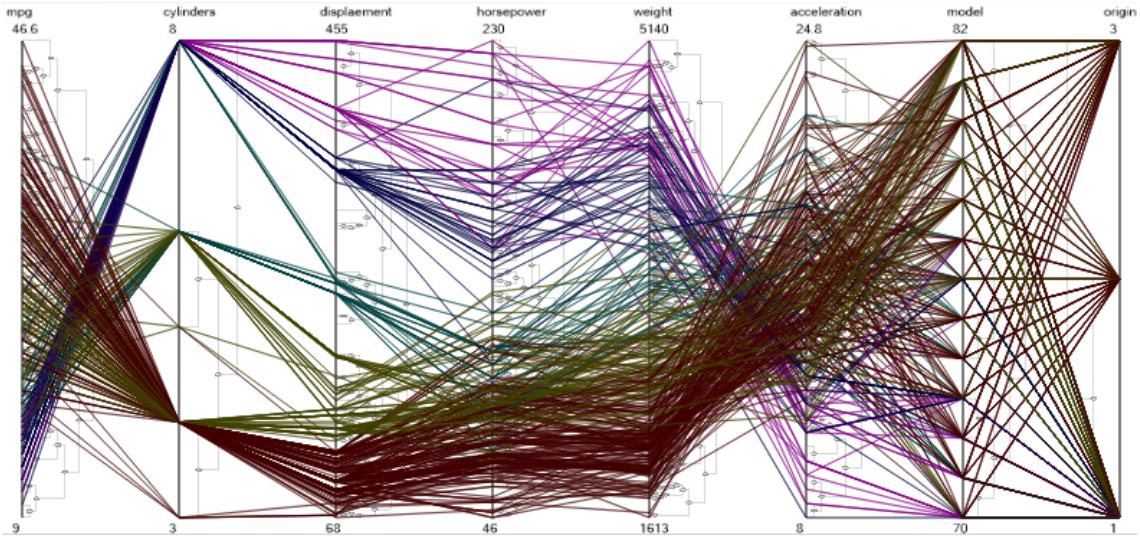


Figure 5.4. Brushing task via the HVN.

The overplot issue is always challenging especially when attempting to brush a large dataset in parallel coordinates. For example, a rendered geometric primitive can be drawn again with a different color in a subsequent rendering and then it eventually ceases to exist. However, such problems exist in almost all visualizations and not just parallel coordinates. Alternatively, alpha blending is often employed to reveal the density.

Alpha blending [80] is an image compositing technique and the process involves mixing a source and background color together with a "*blending ratio*" (a.k.a alpha channel) to a destination pixel. The color component of a pixel can be represented by a 32-bit integers (R, G, B, A) where $A \in [0,1]$ holds an alpha channel. Let RGB_s and A_s denote the source color and alpha respectively. The resulting color can be obtained by:

$$\alpha = A_s + A_d \times (1 - A_s)$$

$$RGB = (RGB_s \times A_s) + \big(RGB_d \times A_d(1 - A_s)\big)/\alpha$$

Equation 5.3

Where α must be greater than zero, if not, simply output the black color ($RGB = 0$) in the destination pixel. Figure 5.5 demonstrates an application of alpha blending in our system where it is obvious that the top diagram reveals a heavily over-plotted view but the bottom one has uncovered a major pattern after applying an alpha value of 0.01. Basically, that means some geometric objects are invisible due to insufficient density.

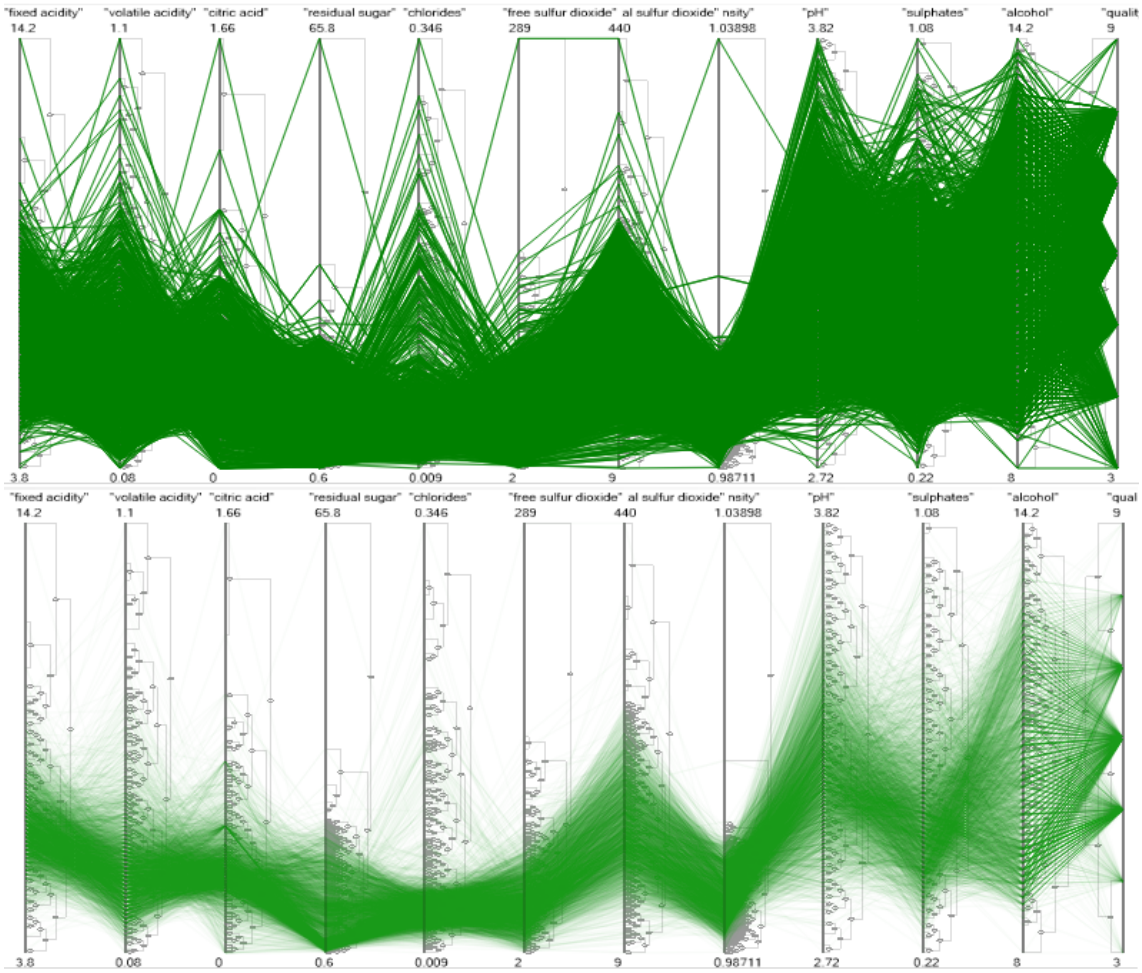


Figure 5.5. Alpha blending for uncovering a major pattern.

One significant drawback of alpha blending is the configuration of the α value is really empirical and more specifically, the data density is relative to the dataset size. For example, Figure 5.6 shows results for α values of 0.7, 0.1, and 0.01 for the top, middle and bottom diagrams respectively. A lower α value eases the overplotting issue but low density patterns tend to be illegible due to high transparency. This may not be desirable if the low density pattern is really statistically significant.

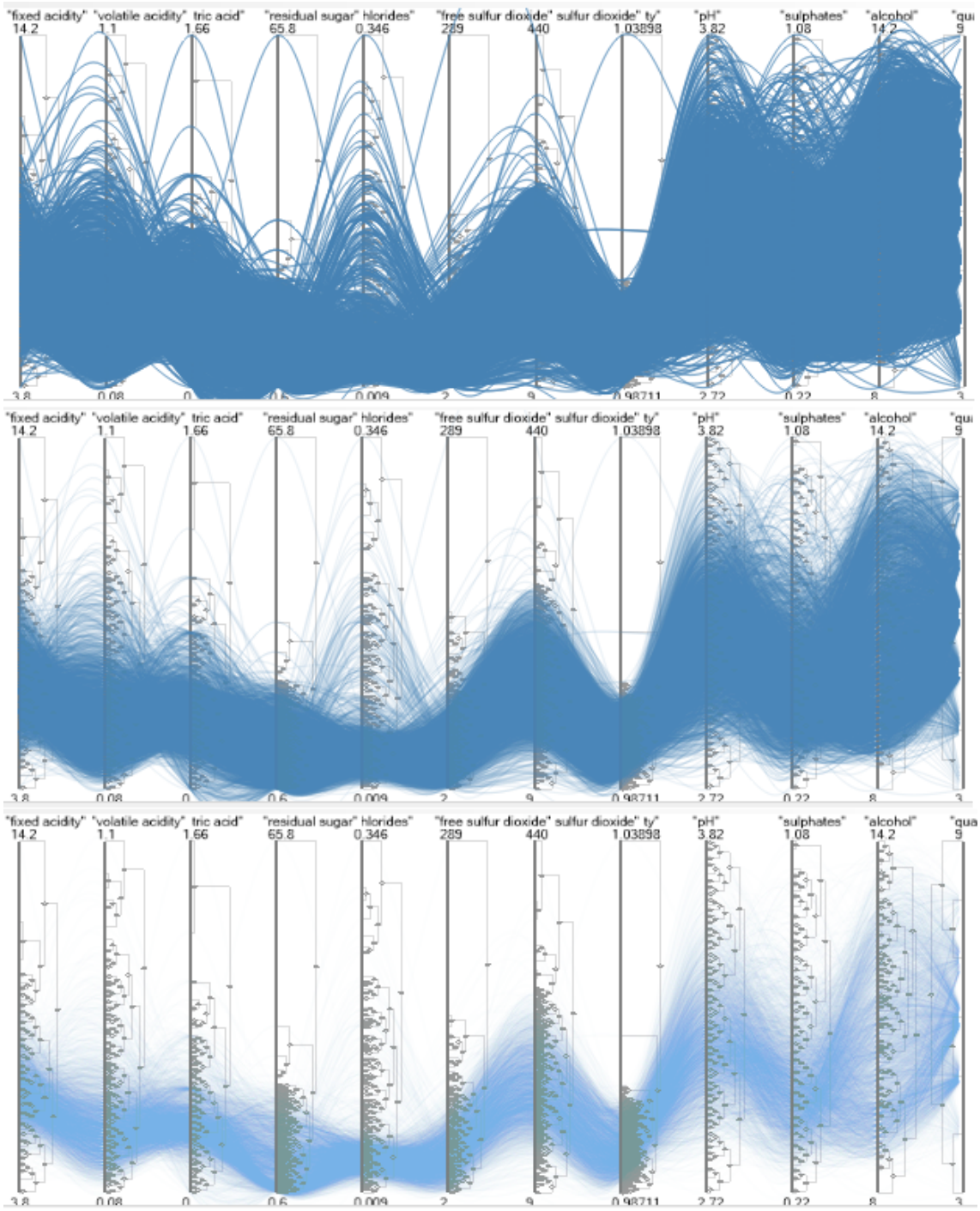


Figure 5.6. Comparison of alpha blending with various alpha values.

5.1.3 Highlighting Detail on Demand

In a typical visual analytics, the user's interest over a subset of data is constantly changing so the rest implies the data noise. *Details on demands* is a feature in our system that allows users to indicate an area of interest to be stand out from the noise.

Details on demands is the last step in the Visual Information Seeking mantra because *details* implies the quality (interested data to highlight) and not the quantity (number of data to highlight). Let (X_{mouse}, Y_{mouse}) and t denote the location of a mouse click and a threshold for the number of data that can be highlighted at a time. Furthermore, let $T = \langle X_i, \Upsilon, \ell_{X_i}, \ell \rangle$ be a tuple that holds 4 elements returned from the $HitTest$ function defined in Algorithm 5.1. The conditional function defined below returns a Boolean to indicate whether a highlighting operation can proceed or not, by our system.

$$CanHighlight(X_{mouse}, Y_{mouse}) \begin{cases} true & |T[1]| \leq t \\ false & otherwise \end{cases}$$

Equation 5.4

Where $|*|$ means a cardinality and $T[1]$ is an indexing operation for the second element in tuple T. The activation of the *details on demand* task slightly differs from data selection. The user needs to hover a mouse cursor over an interested virtual node for activating this task. Figure 5.7 provides a screenshot for the task taken from our system.

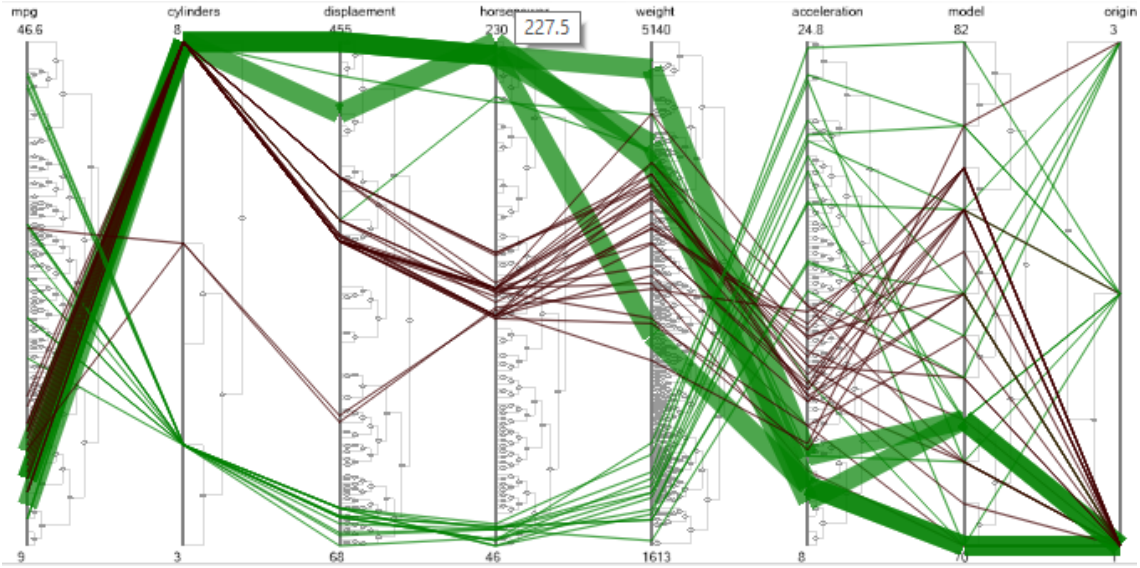


Figure 5.7. An application of detail on demand.

5.1.4 Discussion

A key point that we keep highlighting throughout this work is that, the HVN changes the way to interact with data in parallel coordinates. All the tasks described in this section only require a mouse click or hovering directly over a virtual node. In summary, the HVN is really an efficient and intuitive way to select data in our empirical study that would otherwise have been cumbersome by manipulating through a separate widget.

5.2 Task by Dynamic Viewing

The dynamic viewing layer that merges layers 3, 4 and 5 of Yi's model, allows users to change the way of data representations for achieving better readability or understanding of the data and its relational structures. In the following sections, we will discuss the local and global drill-down to achieve the *details on demand* with case studies. In addition, an analytics task of probability density estimation for the visual analytic will also be introduced.

5.2.1 Hierarchical Local Drill-Down

To the best of our knowledge, the term *drill-down* was first referenced in parallel coordinates by Fua et al. [35] in 1999, and is described as:

> *A process of viewing data at a level of increased detail*.

Zooming is one aspect of drill-down and probably the simplest zooming is *classic zooming* which scales all the graphic objects proportionally with respect to a zooming factor. For advanced zooming, Stolte et al. [81] described a technique of multiscale which is capable of displaying multiple zooming paths for both data and visual abstraction. Multiscale is particularly useful for exploring multiple hierarchies simultaneously. For a great taxonomy of zooming techniques, one can refer to a study conducted by Cockburn et al. [82].

Local drill-down enables users to scrutinize interested subsets of data and our technique is similar to multiscaling by changing the context of a target variable while rests remain fixed. The local drill-down operation is again tightly integrated with the HVN. A conceptual illustration of local drill-down is provided in Figure 5.8 where a brown node indicates a selected virtual node.

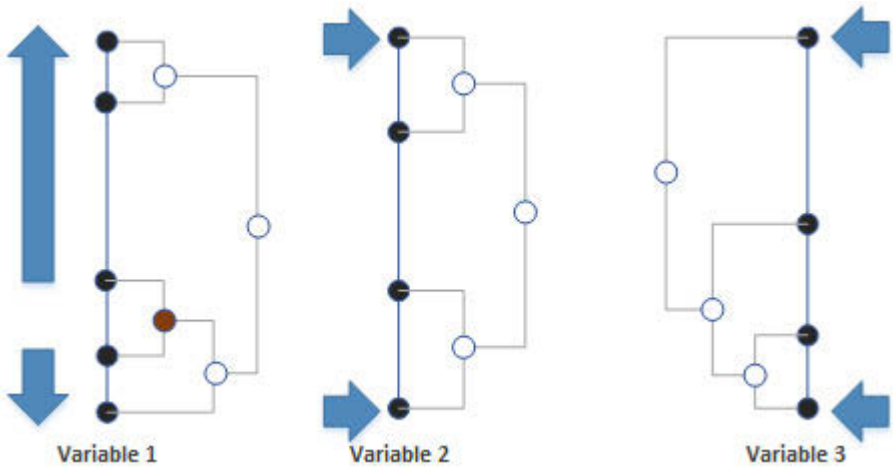


Figure 5.8. Hierarchical local-drill-down. Local drill-down is achieved by changing the maximal and minimal range of a target variable and rests are fixed.

Technically, local drill-down is achieved by triggering the view change for a variable X_i through the adjustment of its maximal and minimal range because all the geometric objects are always positioned relatively to that scale. Recall that a data selection (see Section 5.1.1) returns Υ which holds a list of selected row indices. In addition, the global data matrix D contains a mixture of real and virtual data (see Section 4.3.6.3), so for simplicity, we skip offset computation here and assume i can index a column vector for X_i in D. The procedure of a local drill down is therefore given in Algorithm 5.2.

1. **procedure** $LocalDrillDown(X_{mouse}, Y_{mouse})$
2. $\langle X_i, \Upsilon, \ell_{X_i}, \ell \rangle = HitTest(X_{mouse}, Y_{mouse})$
3. **if** $|\Upsilon| > 0$ **then**
4. **if** $\ell = 0$ **then**

```
5.          /* Data node is selected so set max and min to the data value. */
6.          max_{X_i} = D[Υ_0, ℓ_{X_i}]
7.          min_{X_i} = D[Υ_0, ℓ_{X_i}]
8.        else
9.          max_{X_i} = max_{j∈Υ}{D[Υ_j, i]} /* Compute the new maximum. */
10.         min_{X_i} = min_{j∈Υ}{D[Υ_j, i]} /* Compute the new minimum. */
11.       end if
12.     end if
13.     LayoutVirtualNodes(v, X_i)
14. end procedure
```

Algorithm 5.2. Local drill-down algorithm.

Where *LayoutVirtualNodes* was defined in Algorithm 4.4 and the virtual node can simply be repositioned relative to the new range as depicted in Figure 5.9 where we have reproduced Equation 2.1 for clarity. The logics in our system have been divided into data and visual operations, so we only have to adjust the data values and the drawing procedures defined in Section 4.3.6 will perform the rests such as screen coordinate transformation.

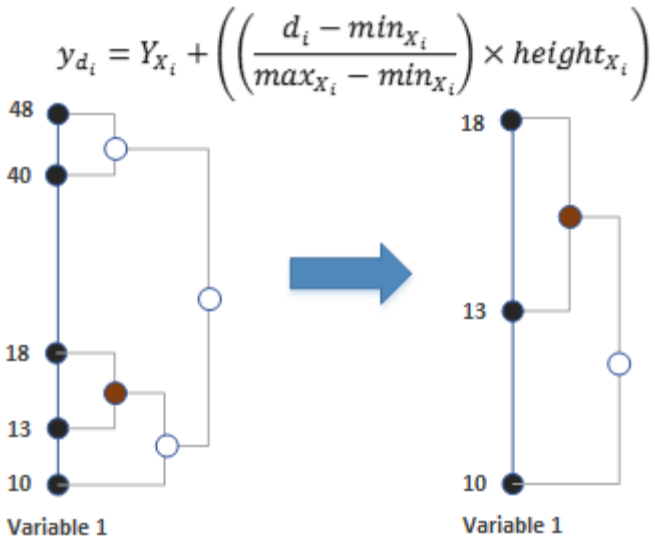


Figure 5.9. Remapping maximal and minimal values in local drill-down.

Local drill-down facilitates the exploration of a multidimensional dataset by allowing users to focus on the details of a variable while other remain fixed. A case study that specifically demonstrates the usefulness of the local drill-down will be presented in Section 7.1.

5.2.2 Hierarchical Global Drill-Down

Global drill-down is similar to local drill-down but instead of adjusting the numerical span for a variable, global drill-down adjusts the ranges for all the variables. It can be easily extended based on the detail described in Section 5.2.1. Global drill-down is suitable for rapid pattern discovery where the user just wants to focus on a view with full respect to numerical ranges from the selected data subset. A case study that specifically demonstrates the usefulness of the global drill-down will be presented in Section 7.2.

5.2.3 Probability Density Estimation

Probability density estimation is an advanced task exposed in our dynamic viewing layer which also corresponds to *Encode* layer in Yi's [14] seven-layer of visual interaction. In visual analytics, many problem domains involve the visualization of density estimation given a random variable. The term *estimation* is used in such a way that datasets were often collected on the basis of finite observations, that is, they are a small subsets of an entire population. Recall that, even though virtual nodes provide the density distribution they do not give smoothing nor estimation.

Histogram (see Section 4.3.7) is the simplest method of plotting data density but the artifact is largely dependent on the bin width and therefore the distribution can be artificially distorted due to a poor bin width chosen. Alternatively, kernel density estimation is often adopted.

Kernel Density Estimation (KDE) [83] [84] is probably the most popular nonparametric method for probability density estimation in many scientific applications. Let X_i be a random variable, its probability density can be estimated by the kernel density estimator introduced by Rosenblatt [83] as:

$$\hat{f}(d) = \frac{1}{Nh}\sum_{i=1}^{N} K\left(\frac{d - D_i}{h}\right), \forall D \in X_i$$

Equation 5.5

In a more generalized form, it is often expressed as:

$$\hat{f}(d) = \frac{1}{N}\sum_{i=1}^{N} K_h(d - D_i)$$

Where $K(x)$ is a univariate kernel function and *d* denotes the data value to be estimated. *h* is the smooth parameter or also known as *bandwidth* that controls the smoothing. For multivariate kernel density estimation, Equation 5.5 can be rewritten as

$$\hat{f}(d) = \frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{M} h_j^{-1} K\left(\frac{d - D_{ij}}{h_j}\right)$$

Where *j* refers to the subscript of the target variable in the multidimensional dataset. A kernel is a weighted function and the choosing of bandwidth *h* is crucial to the shape of the function. In general, the properties of a kernel function should be symmetric around zero and integrating to one as described by Hardle and Linton [85]:

$$K(x) = K(-x); \int K(x)\, dx = 1$$

Where $K(x) \geq 0$. Therefore, any functions satisfying these properties can be regarded as kernel functions in KDE. In general, there are six common kernels [85] and we have reproduced them in Table 5.1.

Kernel	$\boldsymbol{K(d)}$
Epanechnikov	$^3/_4\,(1 - d^2)$ for $\lvert d\rvert \leq 1$
Quartic	$^{15}/_{16}\,(1 - d^2)$ for $\lvert d\rvert \leq 1$
Triangular	$(1 - \lvert d\rvert)$ for $\lvert d\rvert \leq 1$
Gauss	$2\pi^{-1/2} exp\left(-^{d^2}/_2\right)$
Uniform	$^1/_2$ for $\lvert d\rvert \leq 1$

Table 5.1. Six common kernel functions. The table content is based on [85].

The Gaussian kernel with zero mean and unit variance is the most popular kernel and it gives more weight to these data $D \in X_i$ close to d than those away from it. Gaussian is also the kernel that we have applied in the algorithm.

Bandwidth selection The choice of bandwidth h is crucial to the shape of the density estimation. The study conducted by Silverman [86] had also shown that the choice of a kernel does not significantly influence the degree of smoothing but instead it is largely controlled by the bandwidth. The smaller bandwidth produces smaller bin width which implies acute variance with reduced bias and vice versa. Let h_{opt} denote the optimal bandwidth, Figure 5.10 shows the comparison of smoothing produced by various bandwidths as h_{opt} (blue), $h_{opt}/4$ (red) and $4 \times h_{opt}$ (green).

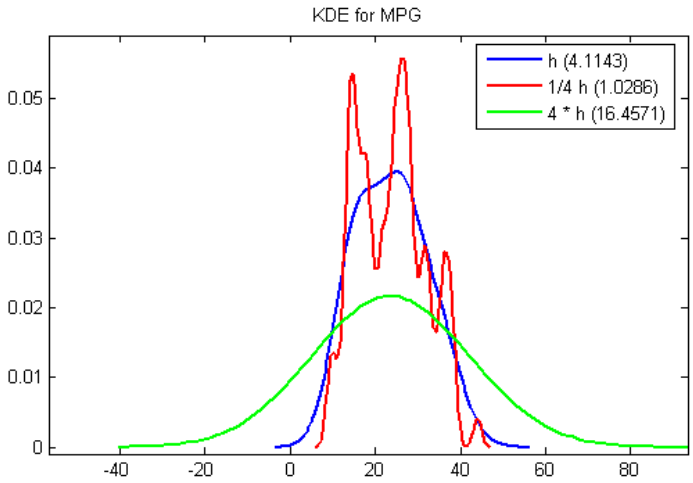


Figure 5.10. Comparison of different bandwidth selection in KDE. The settings of bandwidth are h_{opt} (blue), $h_{opt}/4$ (red) and $4 \times h_{opt}$ (green). The MATLAB function *ksdensity* was used with the bandwidths above to produce the result where h_{opt} is computed by *ksdensity*.

If we choose $h = h(N)$ as a function of N which denotes the number of samples. Parzen [84] has shown us that the expected value of $\widehat{f_h}(x)$ is:

$$E\left(\widehat{f_h}(x)\right) = \frac{1}{hN}\sum_{i=1}^{N}\frac{1}{h}E\left(K\left(\frac{x - X_i}{h}\right)\right) = \frac{1}{h}\int K\left(\frac{x-y}{h}\right) f(y)\, dy$$

Equation 5.6

As we know that KDE is always biased, the common way of choosing a bandwidth is to minimize the Mean Integrated Square Error (MISE) [83] that is given as:

$$MISE\left(\widehat{f_h}(x)\right) = E\int\left(\widehat{f_h}(x) - f(x)\right)^2 dx$$

Where $\widehat{f_h}$ denotes the kernel estimate. By introducing integrated bias and variance terms, the above MISE can be rewritten as:

$$MISE\left(\widehat{f_h}\right) = \int VAR\left(\widehat{f_h}(x)\right)dx + \int Bias\left(\widehat{f_h}(x)\right)^2 dx$$

Equation 5.7

Where *Bias* is a Taylor series expansion of density estimation expressed as:

$$Bias\left(\widehat{f_h}(x)\right) = \frac{h^2\mu_2(K)f''(x)}{2} + o(h^2)$$

Equation 5.8

Where *K* is a kernel function. Similarly, *VAR* is given as:

$$VAR\left(\widehat{f_h}(x)\right) = \frac{R(K)f(x)}{Nh} + o\left(\frac{1}{Nh}\right)$$

Equation 5.9

Where

$$R(K) = \int K^2(x)\,dx$$

Equation 5.10

Equation 5.10 is the kernel variance as noted by Wand and Jones [87] . Obviously, one should expect smaller variance when *h* increases from Equation 5.9 which also corresponds to our observation in Equation 5.6 above.

By substituting Equation 5.8 and Equation 5.9 back to Equation 5.7, Parzen [84] has shown that the bandwidth *h* that minimizes MISE can be written as:

$$h_{MISE} = \left(\frac{R(K)}{R(f'')\mu_2(K)^2}\right)^{1/5} N^{-1/5}$$

Equation 5.11

Where h_{MISE} holds the optimal bandwidth. One problem of Equation 5.11 is that it cannot be used directly because it contains an unknown term $R(f'')$ which measures the speed of the curvature. To address this issue, there are many methods that have already been developed such as plug-in [88] and cross-validation [89]. The rules of thumb [86] is one of the pug-in techniques and we have applied as the bandwidth selector since our kernel is Gaussian.

In the rules of thumb, the unknown term $R(f'')$ of h_{MISE} is replaced by a reference known as the normal distribution for the Gaussian kernel and by substituting it into Equation 2.1. Silverman [86] had also shown that it is reduced to:

$$h_{MISE} = \left(\frac{1/2\sqrt{\pi}}{3/8\,\pi^{-1/2}\sigma^{-5}} \right)^{1/5} = 1.06\sigma N^{-1/5}$$

Equation 5.12

The implementation of KDE with optimal bandwidth is provided in Algorithm 5.3.

1. **function** $KDE(K, d, X_i)$
2. // K – kernel function pointer.
3. // d – the data point to be estimated.
4. // X_i – the univariate variable.
5. **begin**
6. $h = 1.06 \times \sigma_{X_i} \times |X_i|^{-0.2}$
7. $ksum = 0$
8. /* Iterate through each data point. */
9. **foreach** $D \in X_i$ **do**
10. $ksum = ksum + K\left(d - d/h\right)$
11. **end**
12. **return** $\left(ksum/|X_i| \times h\right)$
13. **end**

Algorithm 5.3. Implementation of KDE. X^{σ} and $|X|$ denote the standard deviation and cardinality with respect to the target variable X. The bandwidth is chosen based on the optimal bandwidth.

Figure 5.11 is composed of the screenshots for various bandwidth studies taken in our system. It shows the comparison of smoothing using multiple bandwidths where one can observe the strong variance from the left-most variable which has the smallest bandwidth setting.

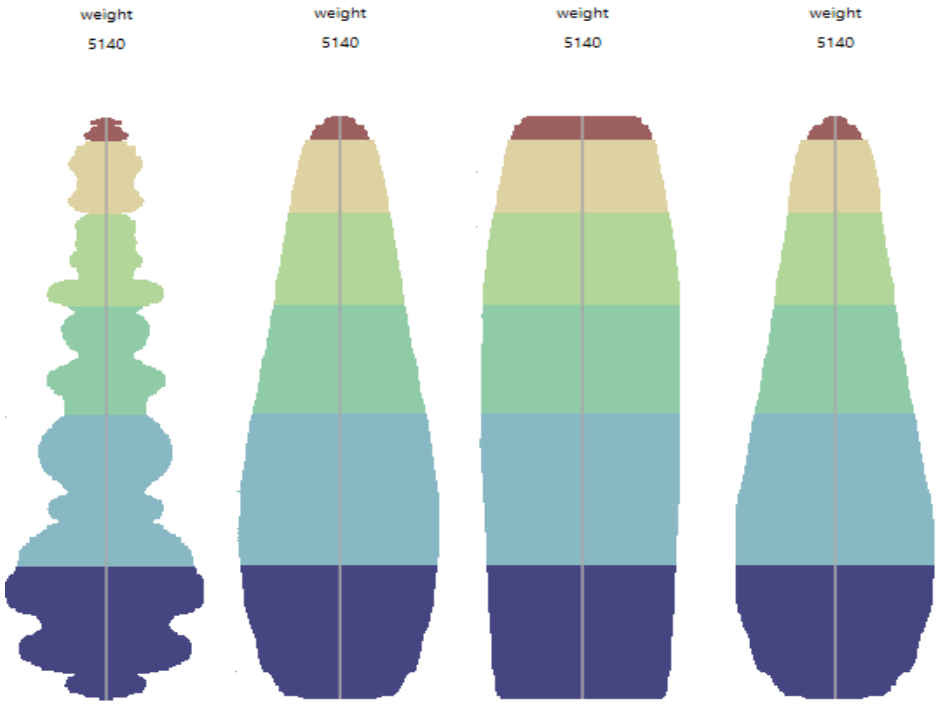


Figure 5.11. Gaussian kernel with various bandwidths. The bandwidth settings are 0.01, 0.1, 0.5 and Gaussian h_{MISE} from left to right.

5.2.4 Variable Overview of Big Dataset

This section is added due to the growing attention on big datasets in information visualization. Visualizing a multidimensional dataset is challenging, visualizing a big dataset is much more challenging. In 1996, the internet was starting to thrive when Shneiderman [56] proposed the Visual Information Seeking mantra. The *overview first*

guideline as part of the mantra profoundly influenced the visualization design. Nowadays, data are produced exponentially so it is not uncommon to deal with a dataset with more than 100 variables. Researchers who deal with big datasets which simply follow the classic *overview first* will soon fail to align with user experience.

To illustrate the complexity, we have visualized the National Youth Tobacco Survey 2009 (NYTS) dataset in classic parallel coordinates. The dataset surveyed high school youths about their attitudes, beliefs, behaviors and influences in relation to the tobacco. It contains 116 dimensions (including metadata) and 22,679 data rows with approximately 2,630,764 data points. Unfortunately, the result was frustrated which presents no useful pattern other than visual clutter as shown in Figure 5.12.

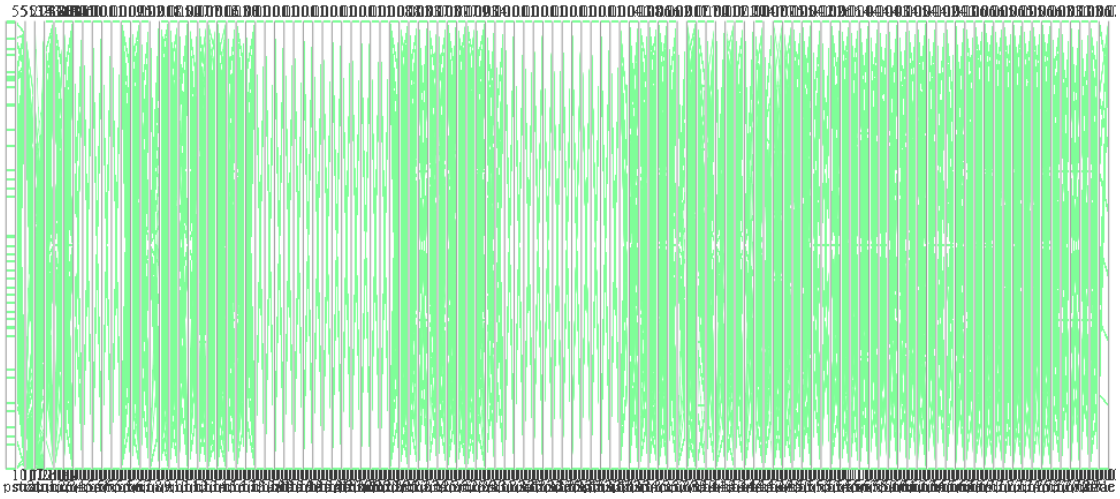


Figure 5.12. Visualization of NYTS 2009 dataset in parallel coordinates. The dataset contains 116 dimensions and 22,679 data rows with approximately 2,630,764 data points. Please refer to Table 4.4 for the dataset source.

The consequences are the cluttered view with struggling system performance that could merely achieve less than 1 FPS approximately. Please note that, we have not yet found a case study of a similar scale through a courtesy scan of relevant literatures. However, we noted a technique called Circle Segments [90] which provided an overview of 50 dimensions with 265,000 data items as shown in Figure 5.13, but it was not designed for an interactive visualization. The scale of the dataset applied in Figure 5.13 is still far less than the one applied in Figure 5.12.

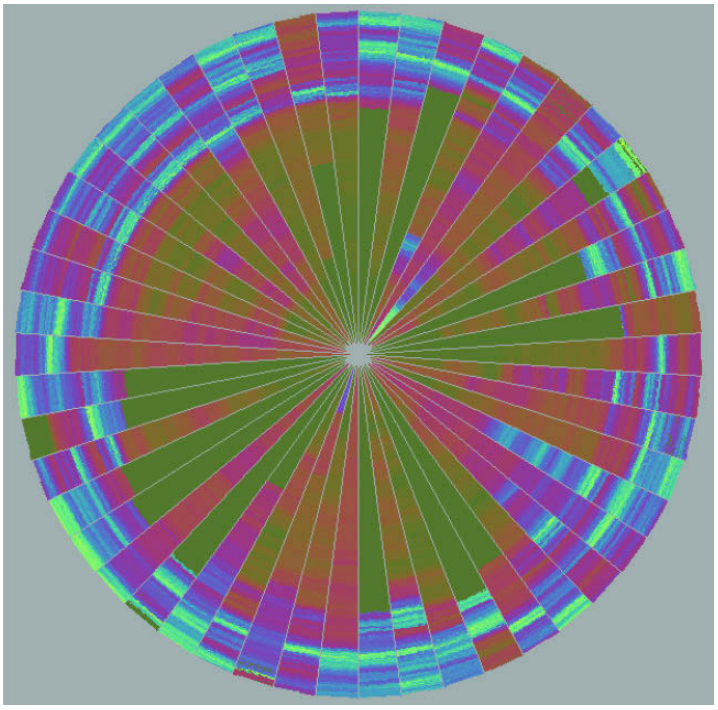


Figure 5.13. Circle segments visualization. The image is sourced from [90].

5.2.4.1 Divide-and-Conquer Model

In this section, we will discuss the divide-and-conquer model for dealing with big datasets in visual analytics. The idea is not new and a similar model in distributed data engineering is called MapReduce which aggregates a massive amount of distributed data into a smaller set for analysis. They are conceptually similar in such a way that they attempt to tackle a problem by breaking down complexities. The basic idea behind the approach of divide-and-conquer is based on several prior works which will be described below.

Liu et al. [55] presented a study about the mental model, visual reasoning and interaction in visualization. The key point learnt with the relevant phrases is quoted below:

When asked about the relative location of San Diego with respect to Reno, people incorrectly responded that San Diego was west of Reno. ……. People do not remember the locations of cities. Instead they remember the relative locations of the states and infer the locations of the cities from the state superset.

Liu reminds us that people are not concerned with detailed aspects of data when information space is large and instead, they tend to learn from its superset. Inspired by Liu's study, the *overview first* is further amended as the *variable overview first* by the use of a correlation matrix as the information superset.

In 2005, van Wijk [8] gave a simple model of visualization in his work titled "*The value of visualization*". In 2008, Keim et al. [9] proposed a sense-making loop based on it. For interactive visual analytics of big datasets, we further develop a divide-and-conquer model based on our framework discussed in Chapter 3 (based on Yi's [14] framework), a simple model of visualization [8], a sense-making loop [9] and also the study contributed by Liu [55].

The model is illustrated in Figure 5.14 which is similar to the *simple model of visualization* as shown in Figure 1.2 and *sense-making loop* conceptually with the main difference being, a divide-and-conquer approach. Variables are "*divided*" by their correlations with color coding for guiding the user to "*conquer*" them. Dealing with hundreds or even thousands of variables, user is often challenged by the question of "*how to start dealing with it*". Therefore, the correlation matrix is designed to shield the user from information overload while providing a sufficient visual hint for the user to start with an interactive visual analytics.

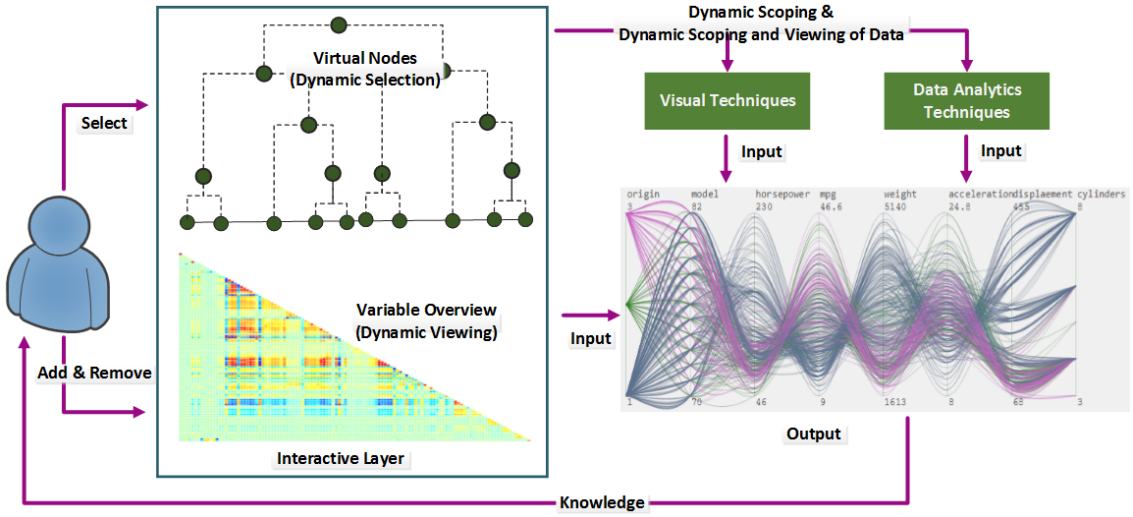


Figure 5.14. An interactive divide-and-conquer model. The model is designed for the interactive visual analytics of big datasets.

5.2.4.2 Overview by Correlation Matrix

The multivariate correlation matrix view is designed to gain the variable insight. It is an auxiliary view and not an integral part of parallel coordinates. The view provides an inter-correlation between variables so the user can interactively add an interested variable to parallel coordinates progressively since it is impractical to place a large quantity of variables simultaneously.

There many ways to measure the correlation and one of the most commonly used methods is Pearson's [91] product moment correlation written as:

$$cor(X,Y) = \frac{\sum_i^N (X_i - \bar{X})\,(Y - \bar{Y})}{\sqrt{\sum_i^N (X_i - \bar{X})^2}\sqrt{\sum_i^N (Y - \bar{Y})^2}}$$

Equation 5.13

The output of $cor(X,Y)$ is in the range between $[-1,1]$ where a positive value means a positive correlation and vice versa. The multivariate distance matrix can be constructed by passing a pairwise variables X_i and X_j to Equation 5.13 at element $[i,j]$ as:

$$D^N = \begin{pmatrix} cor(X_1,X_1) & \cdots & cor(X_1,X_N) \\ \vdots & \ddots & \vdots \\ cor(X_N,X_1) & \cdots & cor(X_N,X_N) \end{pmatrix}$$

From the properties of covariance, we know that the operation $cor(X,Y) = cor(Y,X)^T$ is symmetric. For diagonal elements, $cor(X,X)$ is equivalent to $VAR(X)$. Therefore, the computation of the covariance matrix can be reduced to the tri-diagonal matrix for efficiency. Given a set of variables $X = \{X_1, \ldots, X_N\}$, the correlation matrix can be obtained by passing X to Algorithm 5.4 as below.

1. **procedure** *CorrelationMatrix*$(X = \{X_1, \ldots, X_N\})$
2. **for each** $X_i \in X$
3. **for each** $D_i \in X_i$
4. /* We need to first normalize the values here. */
5. $\widehat{D_i} = \left(d_i - \overline{X_i}\right)/\sigma_{X_i}$
6. **end if**

```
7.    end if
8.    for i ≔ 0 to i < |X|
9.       for j ≔ 0 to j < i
10.         if i ≠ j then
11.            r = cor(X_i, X_j)
12.            color = GetColor(r)
13.            /* Fill the cell for i^th row and j^th column. */
14.            FillCell(color, i, j)
15.         end if
16.      end if
17.   end if
```

Algorithm 5.4. An implementation of the multivariate correlation matrix.

The user interface is a grid layout. It divides the space into grid cells where each cell is color brushed to convey the linear dependency between pairwise variables. The color model applied for denoting the correlation is RGB ramping with hot-cold colors in a sequence of *red*, *red-yellow*, *green*, *blue-green* and *blue* where red and blue represent highly positive and highly negative respectively.

An application of Algorithm 5.4 is provided in Figure 5.15 where the user can interactively add interested pairwise variables to the visualization by clicking on a cell.

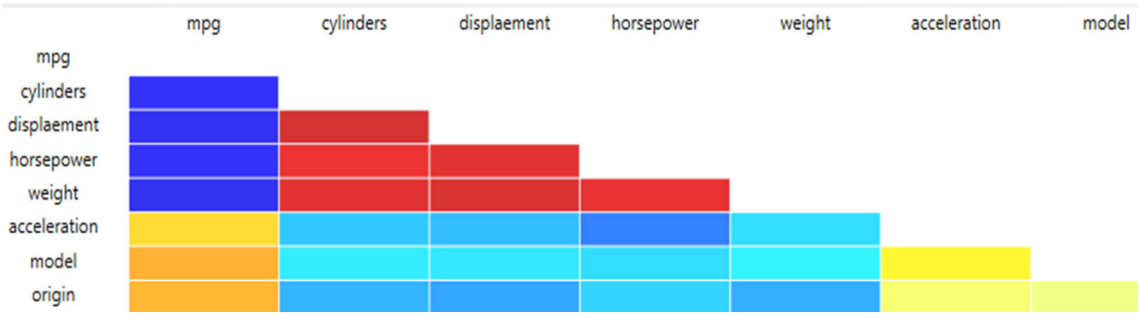


Figure 5.15. Multivariate correlation matrix view of a car dataset.

The scale of the dataset used in Figure 5.15 is probably trivial and not representative. In order to demonstrate the effectiveness of the technique for guiding a

user to explore a big dataset, Figure 5.16 presents a multivariate correlation view of the NYTS 2009 dataset with more than 100 variables. There is no space to fit all the text labels but a dynamic label will show up if one hovers the mouse cursor over a cell.

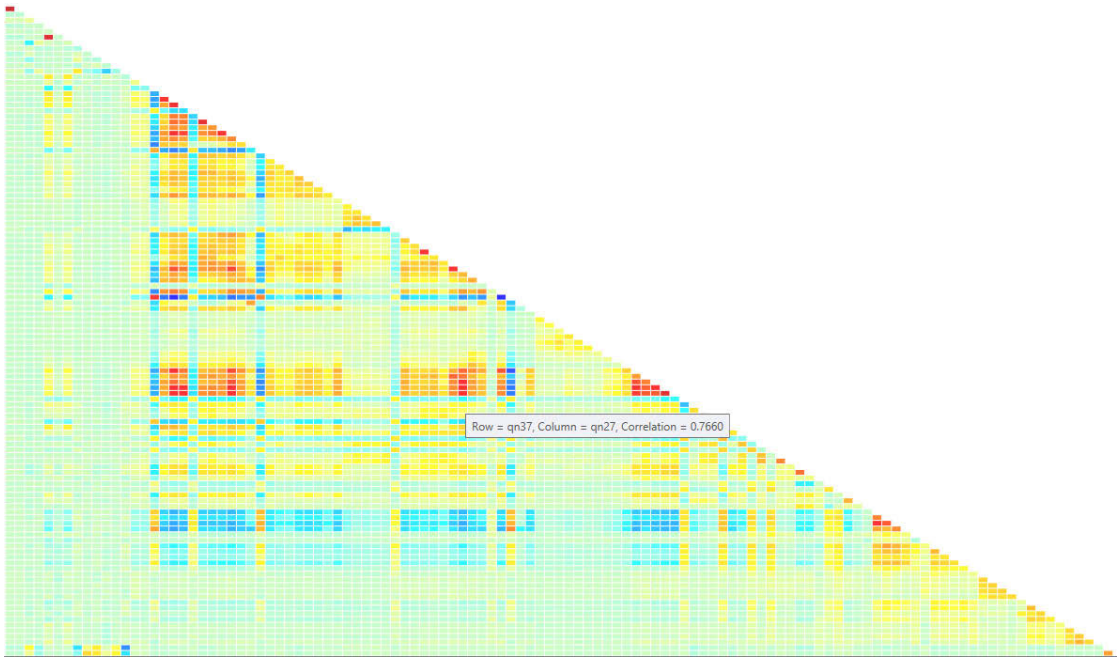


Figure 5.16. Correlation matrix view for the NYTS 2009 dataset.

The novelty of this technique is that, it adds an additional step of the *variable overview* in the Visual Information Seeking mantra to shield a user from cognitive overload. There are some advantages offered by this design. 1) Features are sparse in the high dimensional dataset so it is not necessary to study all the dimensions. The color coding provides a visual hint of understanding the inter-correlation. 2) It allows the user to add or remove a pairwise variables arbitrarily in a divide-and-conquer approach.

The performance is an important consideration in our implementation and the ability to parallelize the computation of correlation coefficient is a primary reason to use a matrix. It decouples the data dependency so elements can be updated concurrently. Figure 5.17 provides a performance measure based on Intel Core i7-3930 with 12 logical processors and 32 GB RAM. It took approximately 2 seconds to build a multivariate correlation matrix for 116 dimensions.

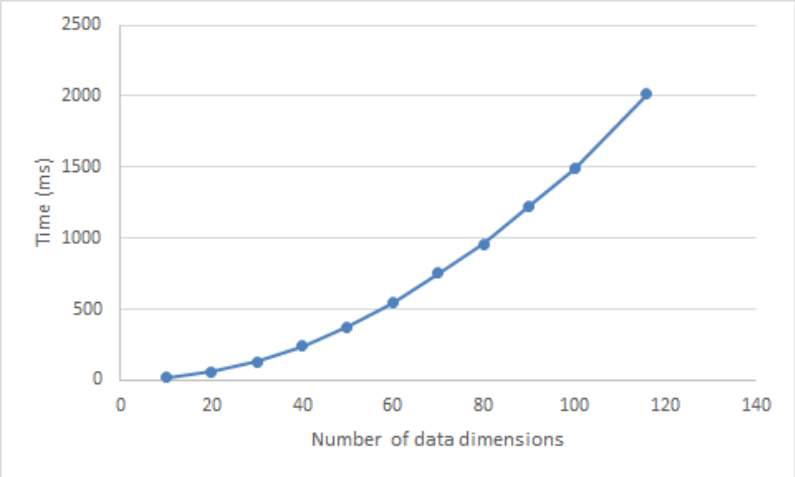


Figure 5.17. Performance of building the multivariate correlation matrix.

Overview by Multidimensional Scaling (MDS) First of all, MDS is not implemented in our system but the primary purpose to describe it here is to document a potential application in the variable overview. MDS is probably well-known for its application in dimensionality reduction. It was first introduced by Torgerson[7] [92] for mapping the distance of dimensional correlation and such capability inspires us to extend its application in guiding a user over the overview presentation of a large multidimensional dataset. The basic idea is similar to the multivariate correlation matrix view discussed in the previous section. For a complete introduction of class MDS procedures, one should refer to [93]. Let $X = \{X_1, X_2, \dots, X_N\}$ be a multidimensional dataset which needs to be converted into a proximity matrix by the following distance measure as:

$$D(X_i, X_j) = \sum_{d_i \in X_i} \sum_{d_j \in X_j} \sqrt{(d_i - d_j)^2}$$

Equation 5.14

Equation 5.14 is essentially a two dimensional Euclidean function. Alternatively, one can use the Pearson correlation in Equation 5.13 to construct the proximity matrix if

[7] There are many variants of MDS but the method originally introduced by Torgerson is known as classic MDS.

the concept of distance is not applicable on the given dataset. The first step in classic MDS is to square the proximity matrix by:

$$X^2 = \begin{pmatrix} D(X_1, X_1)^2 & \cdots & D(X_1, X_N)^2 \\ \vdots & \ddots & \vdots \\ D(X_N, X_1)^2 & \cdots & D(X_N, X_N)^2 \end{pmatrix}$$

Next, apply the double centering as:

$$B = -\frac{1}{2} J X^2 J$$

Equation 5.15

Where *J* is given as

$$J = I - N^{-1}[1]$$

Where *N* is the cardinality of *X*. [1] and *I* denotes the unit matrix of ones and the identity matrix respectively. They are trivially expressed as follows:

$$[1] = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}, I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The Singular Value Decomposition (SVD) is applied on *B* to obtain the first two largest positive eigenvalues $\{\lambda_1, \lambda_2\}$ and corresponding eigenvectors $E_2 = \{e_1, e_2\}$. We are only interested in the first two largest eigenvalues because the final representation is a projection of the two dimensional scatterplot of *N* variables. To work out the two dimensional coordinate matrix M_2, eigenvectors need to be multiplied with the diagonal matrix of eigenvalues.

$$M_2 = E_2 {A_2}^{1/2}$$

Where A_2 holds the diagonal matrix of two eigenvalues. E_2 is the union of two eigenvectors in matrix. The final procedure negates the sign of the coordinate matrix as $-M_2$. The MDS map is obtained by projecting M_2 in a scatterplot. Figure 5.18 provides an example of using MDS for variable overview.

Figure 5.18. Application of MDS map for variable overview. The image was produced in MATLAB.

5.2.5 Discussion

In summary, this section describes several techniques in the layer of dynamic viewing. Local and global drill-down were developed on the basis of the HVN that provide an effortless way of navigating data. We also introduced the use of KDE to study the probability density distribution. Even though, virtual nodes offer the information of data distribution but they do not give smoothing and estimation. Finally, a model and technique have been presented for interacting with big datasets. A case study that demonstrates the effectiveness of the divide-and-conquer model to tackle a big dataset will be presented in Section 7.3.

5.3 Task by Dynamic Scoping

Dynamic scoping (DS) interaction, that merges layers 2, 6 and 7: *Explore*, *Filter* and *Connect* of the interaction are defined in J. S. Yi's model, allowing users to visualize a subset of the data through techniques such as filtering or dimensionality reduction. The technique of filtering is trivial so we will not discuss it here.

5.3.1 Dimensionality Reduction by RST

Dimensionality reduction is considered as an advanced task in visual analytics. A dataset is reduced to a smaller subset while being sufficient to describe a whole set of variables. To approach the dimensionality reduction, the technique adopted here is the Rough Set

Theory (RST) (see Section 2.3.3.2). It offers a distinct advantage over others because of the concepts of *condition* and *decision*. Users simply specify a dimension as a decision and rest become conditions so the dimensions are reduced in such a way that they fully respect to the user specified decision.

Variable Precision Rough Set RST was initially designed to deal with a consistent dataset by its strict definition of approximation regions. It assumes the underlying dataset is consistent and possesses complete certainty in terms of classifying objects into correct approximation regions. For example, if $ab \rightarrow D$ then $cd \rightarrow D$ is considered to be conflicting. This assumption of the error-free classification of the consistent dataset is unrealistic in relation to most real world datasets. Although, a dataset can be partitioned into consistent and inconsistent data space and operates RST on the consistent one, we considered this to be meaningless and impractical for use in this case. To deal with the inconsistent dataset, Ziarko [94] argued that partially incorrect classification should be taken into account and accordingly proposed the Variable Precision Rough Set (VPRS) model as an inconsistent dataset extension to RST. VPRS model allows for probability classification by introducing a precision value β to relax the strict classification in original RST. It introduces the concept of *major inclusion* to tolerate the inconsistent dataset and the definition of *majority* implies no more than 50% of classification error so the admissible range of β is $(0.5, 1.0]$. The β positive in the VPRS model is defined as:

$$POS_P^{\beta}(D) = \bigcup_{\Pr(IND(D)^*|X_i)\geq\beta} \{X_i \in IND(P)\}$$

Where $IND(D)^*$ and X_i denotes a set of the equivalent classes for D and $P \subseteq C$ respectively. Clearly, a portion of objects with specified value β in the equivalence classes need to be classified into the decision class for it to be included in the β positive region. Ziarko also formulated the definition for *quality of classification* that is used to extract the β reducts and we will explain the definition of *reduct* in the next section.

$$\gamma^{\beta}(P,D) = \frac{\left|\cup_{\Pr(IND(D)^*|X_i)\geq\beta} \{X_i \in IND(P)\}\right|}{|U|} \; where \; \Pr(P|D) = \frac{|D \cap P|}{|P|}$$

Where $|*|$ denotes the cardinality for the union of all the equivalence classes in the β positive region where classification is possible at specified the value β with respect to relation $IND(D)^*$ and $|U|$ denotes the cardinality of the universe. Obviously, the quality

of classification provides the measure for the degree of attribute dependency in such a way that if $\gamma^{\beta}(P, D) = 1$ means D fully depends on P at specified β value.

There are certain advantages of using RST over other methods such as PCA, 1) it minimizes the impact of information loss by removing the irrelevant or dispensable dimensions and 2) the resultant subset of attributes is more intuitive by preserving the quality of classification. Typically we may find several subsets of attributes that satisfy the criteria called *reduct sets* denoted as $R = \{P: P \subseteq C\}$. The minimal cardinality in the reduct sets called the minimal reduct denotes as R_{min} where $R_{min} \in C$ is the minimum subset of the condition attributes that cannot be reduced anymore while preserving the quality of classification with respect to the decision attribute. In the VPRS model, the reduct is called β-reduct denoted as $RED^{\beta}(C, D)$ and according to Ziarko a subset $P \subseteq C$ is a reduct of C with respect to D if the following two criteria are satisfied:

1. $\gamma^{\beta}(C, D) = \gamma^{\beta}\left(RED^{\beta}(C, D), D\right)$ and,

2. No attributes can be eliminated from $RED^{\beta}(C, D)$ without affecting the requirement (1).

The requirement (2) can also be mathematically expressed as $POS^{\beta}_{P-\{a\}}(D) \neq POS^{\beta}_{P}(D), a \in P$. Obviously, Ziarko has defined a strict satisfaction of the β reduct in relation to the requirement (1) that some attributes can only be removed if the qualification of classification γ^{β} for subset $P \subseteq C$ is the same against γ^{β} for the whole set of original attributes C.

The applications of the RST have been demonstrated in Figure 5.19 where the decision variables selected were *cylinders* and *experience* for car and wage datasets respectively.

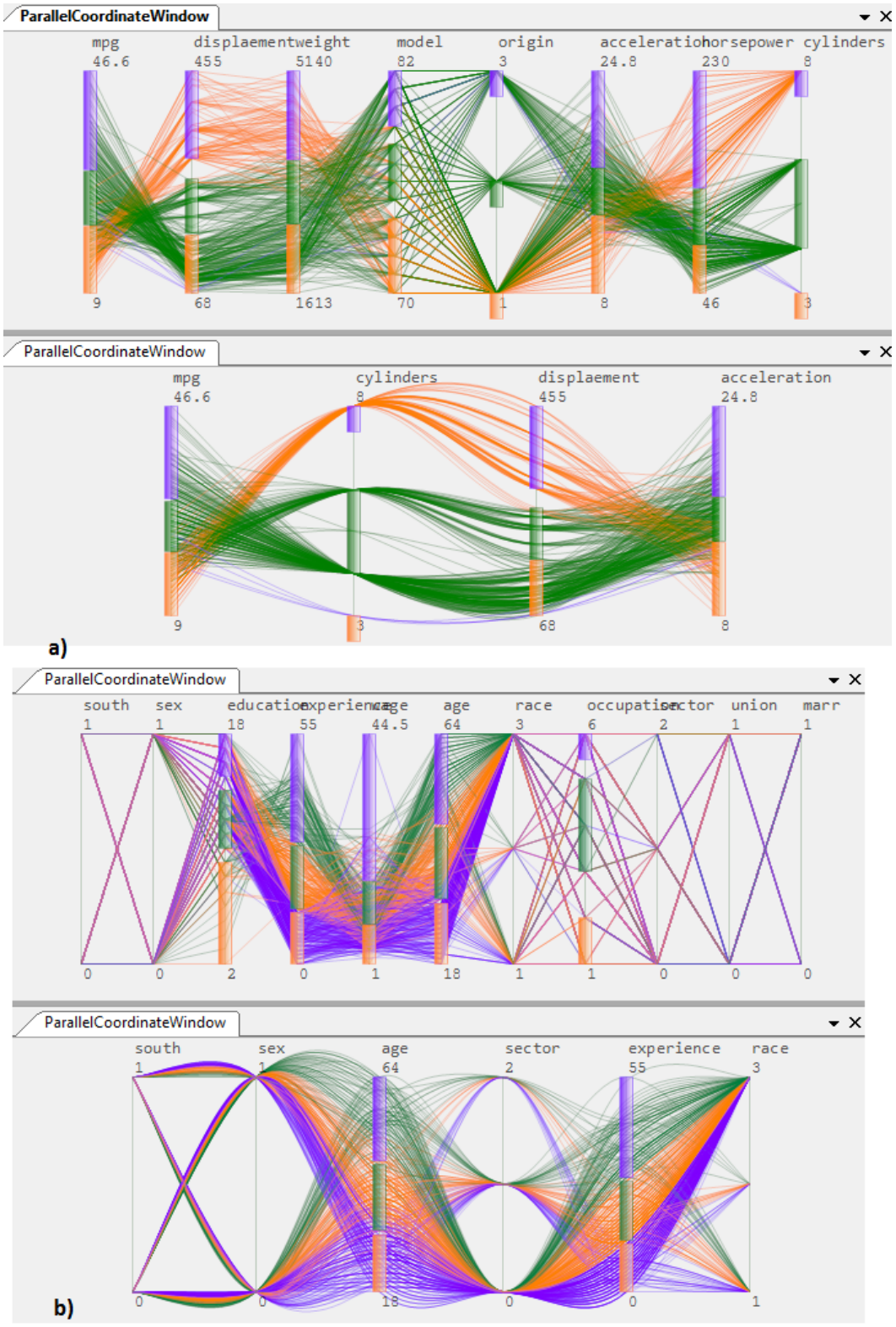


Figure 5.19. Applications of RST. a) The car dataset has been reduced from 8 to 4 variables. b) The wage dataset has been reduced from 11 to 6 variables.

5.3.2 Discussion

This section introduced the use of RST for dimensionality reduction in the layer of dynamic scoping. Many methods of dimensionality reduction exist such as principal component analysis, self-organizing maps or MDS (See Section 5.2.4) but the distinct advantage of RTS over other methods are the concepts of *decision* and *condition* variables. Such concepts can minimize the perception of information loss which is often used to criticize the result if it is not intuitive e.g. a variable that is expected to be retained but has been excluded. To the best of our knowledge, we have produced the first application of RST in parallel coordinates [41] for dimensionality reduction.

Chapter 6 Technical Evaluations

This chapter will present some technical evaluations of our HVN based parallel coordinates visualization against other publicly available visualization systems.

6.1 Visual Clutter of Overview

The versatile HVN allows us to adopt a different approach of overview presentation in parallel coordinates, as already discussed in Section 4.3.7. The central idea is that the organization of virtual nodes already provides the initial data insight of data distribution so that it is redundant to draw classic polylines which often create clutter in many cases. The challenge of evaluating the visual performance comes from the quantization of visual clutter since the term *clutter* is abstract. However, the study conducted by Rosenholtz [95] has suggested two approaches to measure it namely, subband entropy and feature congestion. Subband entropy measures how well the content in an image is organized by first decomposing an image into subbands of wavelength and sums up the entropies computed for each subband. Feature congestion measures the degree of the visual search based on the statistical saliency model. For example, how well an obverse can find a target amongst other graphics objects in an image.

Our evaluation was conducted with 3 datasets and 3 implementations of parallel coordinates and their details are described in the following table.

	Name	**Comment**
Dataset	Car, Wage and Wine	See Table 4.4.
Parallel coordinate visualization 1	Classic PC	This is our implementation of classic parallel coordinate.
Parallel coordinate visualization 2	HVN	This is our implementation of parallel coordinate based on HVN.
Parallel coordinate visualization 3	GGobi	A publicly available software. See [96].

Measurement 1	Subband entropy	MATLAB code was written by Rosenholtz [95].
Measurement 2	Feature congestion	MATLAB code was written by Rosenholtz [95].

Table 6.1. The setup of evaluating visual clutter. The MATLAB source code was obtained from http://dspace.mit.edu/handle/1721.1/37593.

The steps executed in the evaluation are outlined below.

- Produce the images of dataset overview in PNG format for 3 datasets in each parallel coordinate visualization.
- Execute getClutter_SE function in MATLAB and record the output value.
- Execute getClutter_FC function in MATLAB and record the output value.

Where getClutter_SE and getClutter_FC are the MATLAB functions for subband entropy and feature congestion respectively. Figure 6.1 and Figure 6.2 show the results of the visual clutter measurements.

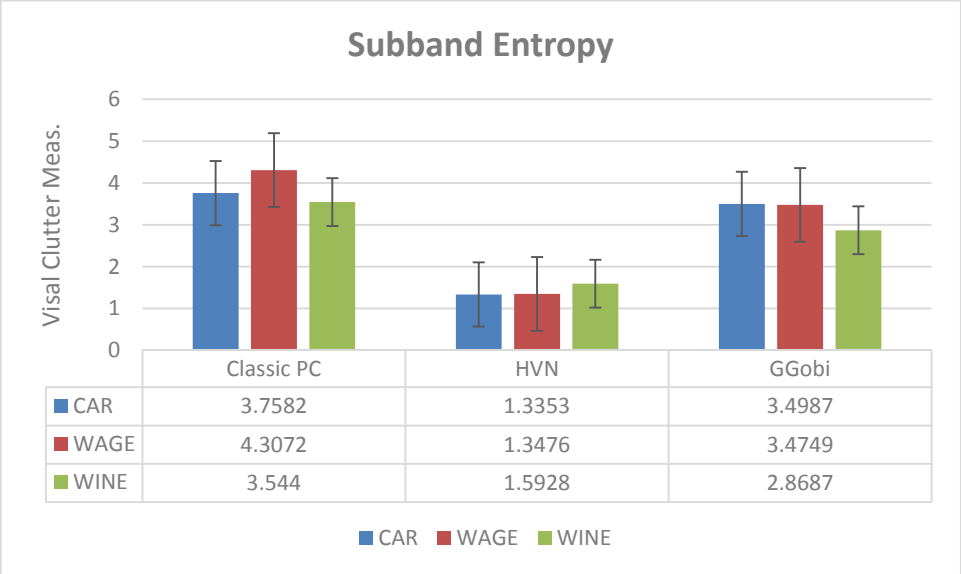


Figure 6.1. Subband entropy measure of visual cutter.

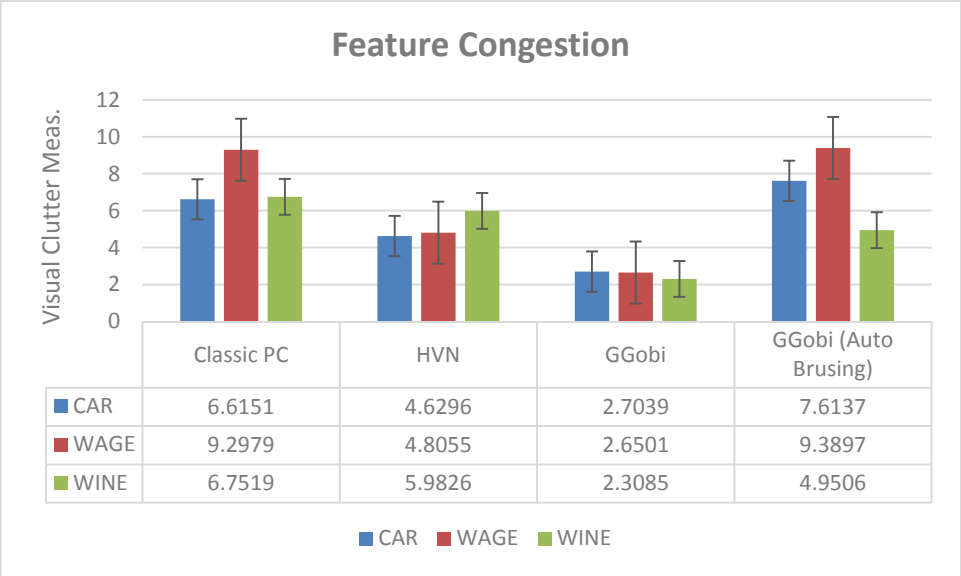


Figure 6.2. Feature congestion measure of visual clutter.

According to Figure 6.1, the HVN has incurred less visual clutter. Interestingly, GGobi outperforms the HVN without the application of automatic brushing in the measurement of feature congestion but the algorithm considers the image as more cluttered with the brushing enabled. This probably suggests that the brushing is not suitable in the overview presentation.

6.2 Data Selection

One of the key innovations that the HVN has made in parallel coordinates is the materialization of the point selection hierarchically and therefore, it is paramount for us to compare its efficiency and accuracy with other techniques. The third-party visualizations that have been chosen in this evaluation are GGobi and Mondrian [60]. The data selection model coincidently incorporated in both is the 2D rectangle (see 4.2.1) with activation over points. That is, the drawing of the rectangular bound over polylines will not activate the data selection mechanism. In this section, we will conduct two common use cases of continuous and non-continuous selection in order to evaluate how well these visualizations support the basic interactivity with data.

6.2.1 Continuous Neighbour Selection

The continuous neighbour selection is a rudimentary evaluation of the select operation. Basically, we try to evaluate the accuracy and the error rate when attempting to mark a continuous range of data as selected directly on the display. Interestingly, GGobi and Mondrian both adopt the 2D rectangle to include data over points which is an elementary technique of interaction since a polyline other end points does not occupy a bounding region from the perspective of a visualization system. Therefore, it is much easier to work out whether a point is enclosed by a rectangle rather than the expensive computation of a point-to-line intercept.

In this evaluation, it was started first by loading the car dataset into GGobi, Mondrian and our HVN. The overview of GGobi and Mondrian is provided in Figure 6.3 and Figure 6.4 respectively. In our experience, we found that the 2D rectangle is cumbersome in terms of aligning the mouse cursor and this is also evident in the figures. Specifically, it might lead to much trial and error because the misalignment of few pixels can lead to unwanted selection if the gap between the continuous data is too small. Accordingly, we decided to carry out the selection on the variable *cylinders* for simplicity because the gaps are apart.

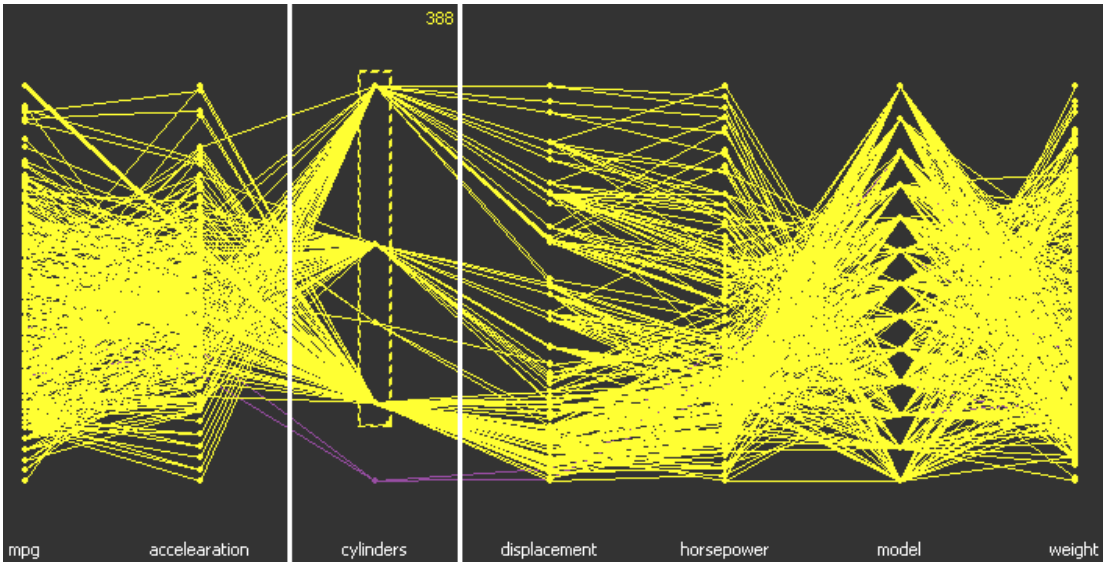


Figure 6.3. 2D rectangular data selection in GGobi.

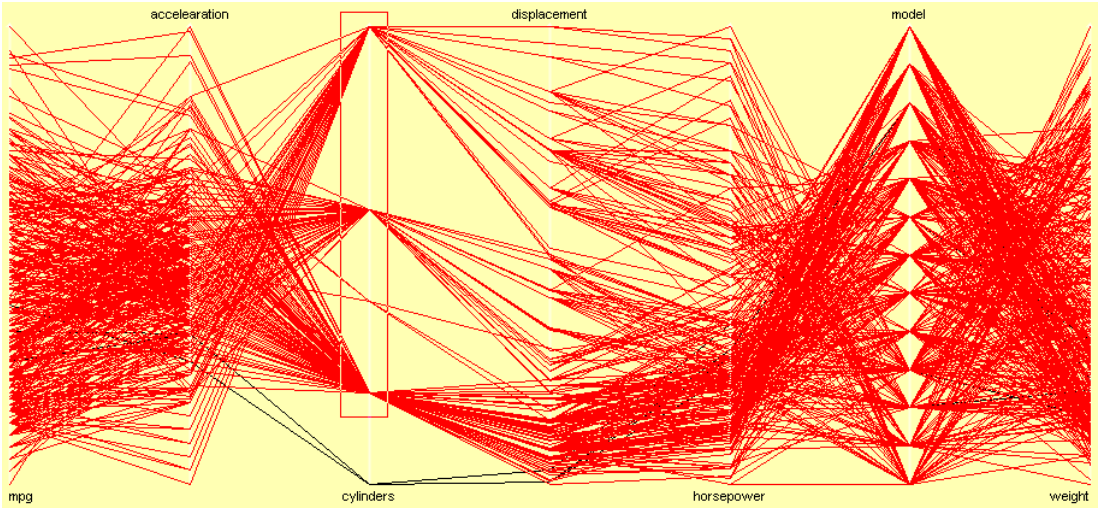


Figure 6.4. 2D rectangular data selection in Mondrian.

The result is recorded in Table 6.2 where all the visualizations were able to achieve 100% accuracy without error.

	Selection Count	Unwanted Data	Selection Model	Error Rate
HVN	388/388	0	Point selection	0%
GGobi	388/388	0	2D rectangle	0%
Mondrian	388/388	0	2D rectangle	0%

Table 6.2. Result of the evaluation of the continuous neighbor selection.

6.2.2 Non-Continuous Selection

In visual analytics, it is a common practice to explore data patterns between groups with diverse quintiles. For example, *Group A* with the value range 1~20 and *Group B* with the value range 65~90. Therefore, we would like to evaluate the facility catered by the visualizations to deal with such use cases. In our HVN, the operation was accomplished by clicking directly on the nodes as indicated in Figure 6.5.

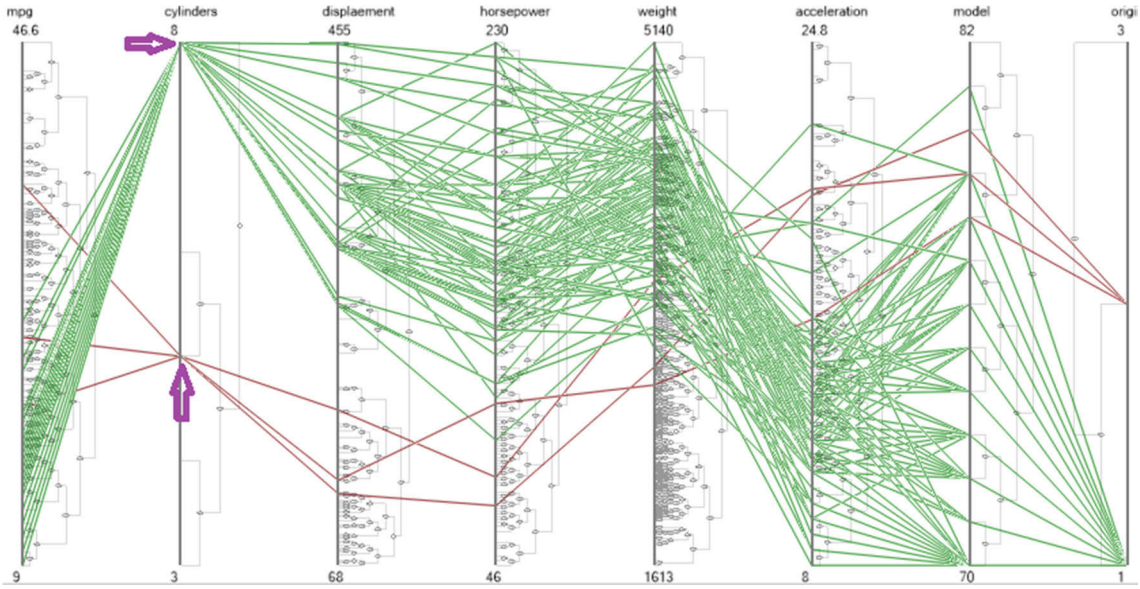


Figure 6.5. Non-continuous data selection in the HVN.

However, it is not straightforward to carry out the same task on GGobi and Mondrian because they only allow a single 2D rectangle so the workaround for us is to coerce the inclusion of undesired data in between. This creates a poor user experience but on the other hand, it highlights the practicality of the HVN as we have successfully used the HVN to achieve the operation that is otherwise impossible to achieve by the others.

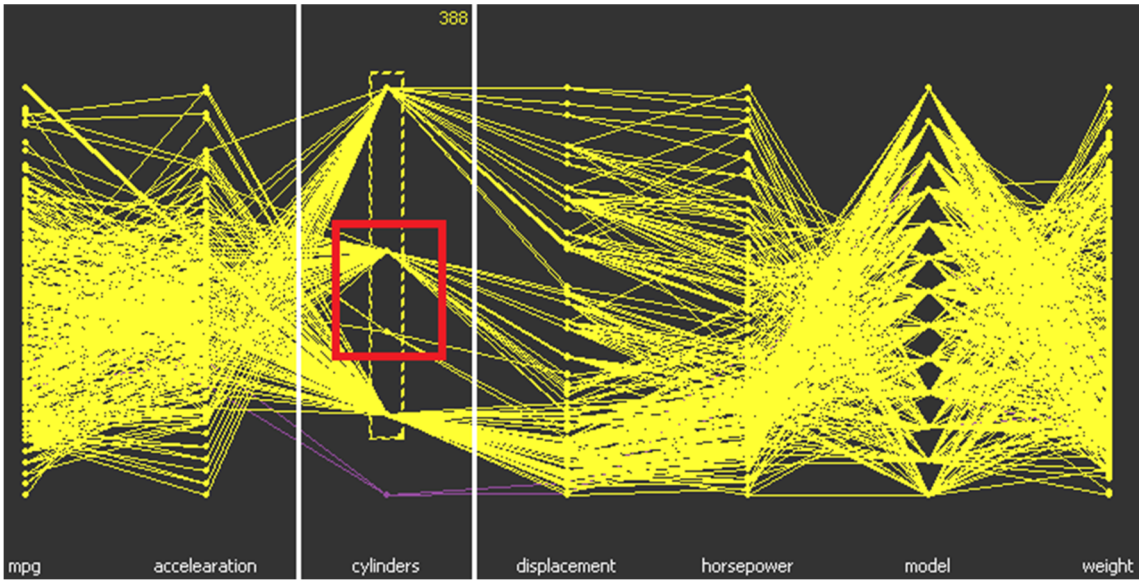


Figure 6.6. Coerce data selection in GGobi.

The evaluation result is provided in Table 6.3 where the HVN outperforms GGobi and Mondrain without the error rate penalized by the data coercion.

	Selection Count	Unwanted Data	Model	Error Rate
HVN	302/302	0	Point selection	0%
GGobi	388/302	86	2D rectangle	28.5%
Mondrian	388/302	86	2D rectangle	28.5%

Table 6.3. Result of the evaluation of the non-continuous selection

To further study the technique of data selection materialized in other parallel coordinates visualizations, we discovered a website [97] that lists approximately 1897 applications of D3.js[8] [98] and 10 of them contain the keywords *parallel coordinates*. We further explored them and were surprised by the fact that the data selection is all designed to be similar to the interface which combines both the value range and 2D rectangle. That is, the 2D rectangle aligns strictly to the vertical axis but this essentially offers no functional difference to GGobi and Mondrian. Figure 6.7 illustrates one of the applications of parallel coordinates visualization in d3.js that we explored.

In summary, we ascertained that the HVN is an innovative technique for data selection in parallel coordinates because it simplifies the cumbersome procedures of activating a data selection into a straightforward mouse click. The same operation that could be easily carried out in the HVN with a higher degree of accuracy is difficult in the others especially, under the circumstance of overplot.

[8] D3.js is a popular JavaScript library that can be used to create a powerful visualization in a data-driven approach.

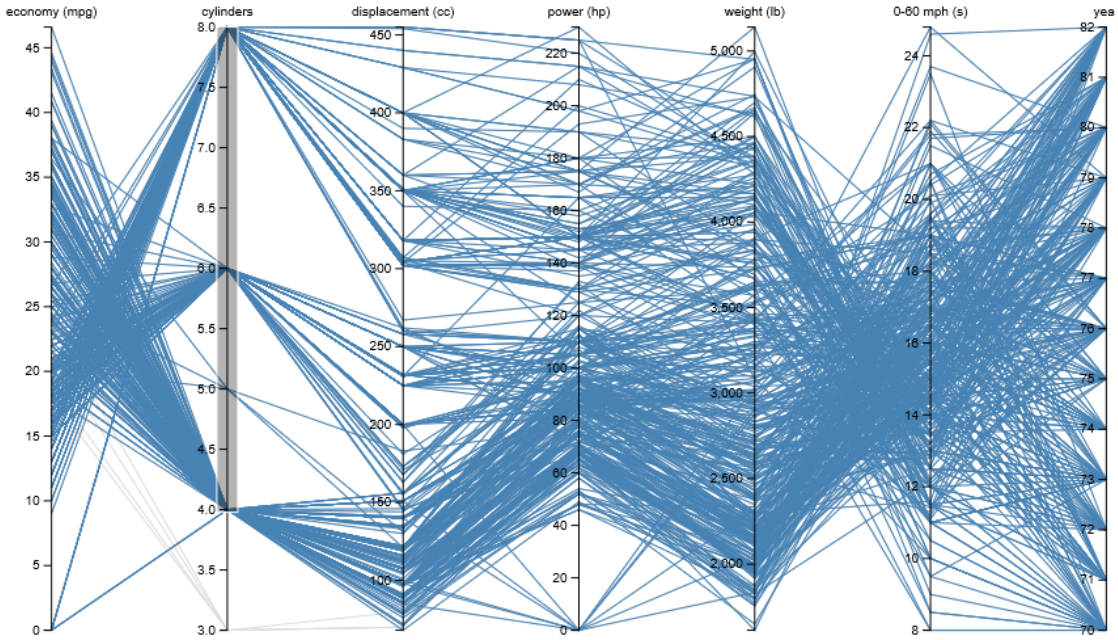


Figure 6.7. An application of parallel coordinates visualization in d3.js. The image is sourced from http://bl.ocks.org/jasondavies/1341281. The website *bl.ocks.org* is run by Mike Bostock.

6.3 Drill-Down

This evaluation is concerned with the usability of the general drill-down facility provided by the visualization appraised. Drill-down provides a means for the dynamic navigation of data and a parallel coordinates visualization reveals that such a well-designed feature can greatly improve the user experience by allowing the user to quickly focus in or out on the area of interest.

We first proceeded with GGobi but there is no way to materialize such an operation directly in the user interface. Thus, we needed to open a separate window and manually entered the values 39 and 46 as the user defined range for the variable *mpg* as highlighted in the top image of Figure 6.8. Unfortunately, the result confused us due to the distortion of the geometric primitives in Figure 6.8. Certainly, from our perspective, the resulting frustration created a poor user experience.

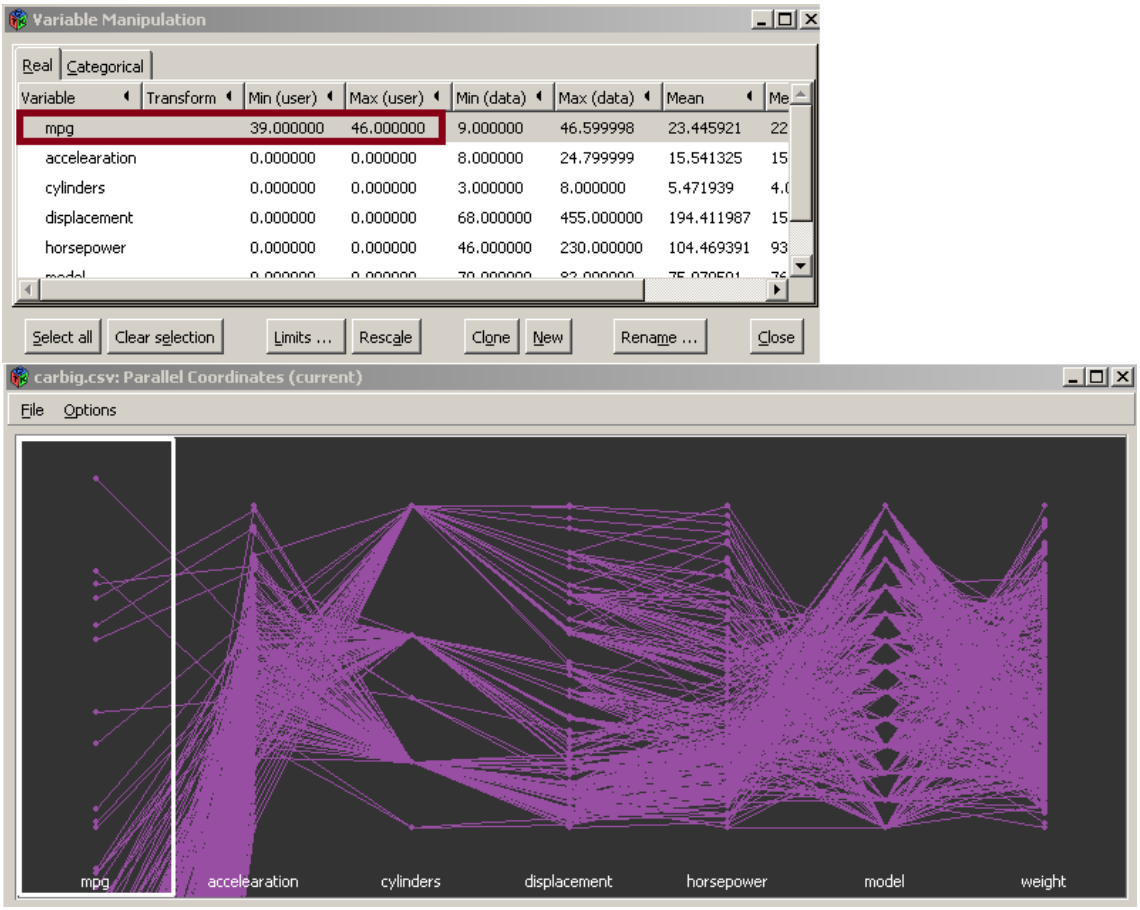


Figure 6.8. Evaluation of drill down feature in GGobi. (Top) Manually enter the user-defined range via a separate window in GGobi. (Bottom) The view which reflects the user-defined range.

The next visualization examined is Mondrian where the holistic tasks of interaction can be accessed by a menu which is activated through the classic approach of the right mouse-click on the user interface. The first related feature explored is called the data alignment but the result is really convoluted as there is misalignment with all the rendered geometric primitives so they are not intuitive for the purpose of interpretation. A screenshot of such an operation is illustrated in Figure 6.9.

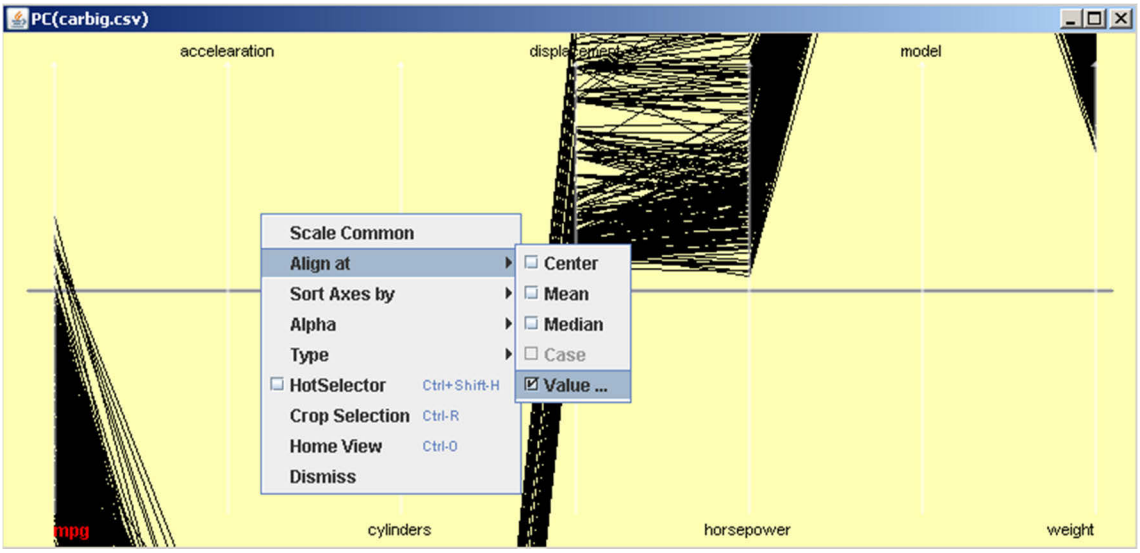


Figure 6.9. Convoluted result of data alignment in Mondrian.

In Mondrian, the function that most aligns to our expectations is probably called the *Scale Common* as shown in the topmost menu item in Figure 6.9. It seems to scale the view by setting a uniform range for all the variables at the global maximum and minimum. This is not very useful when dealing with a multidimensional dataset due to the discrepancies of the variable measurement.

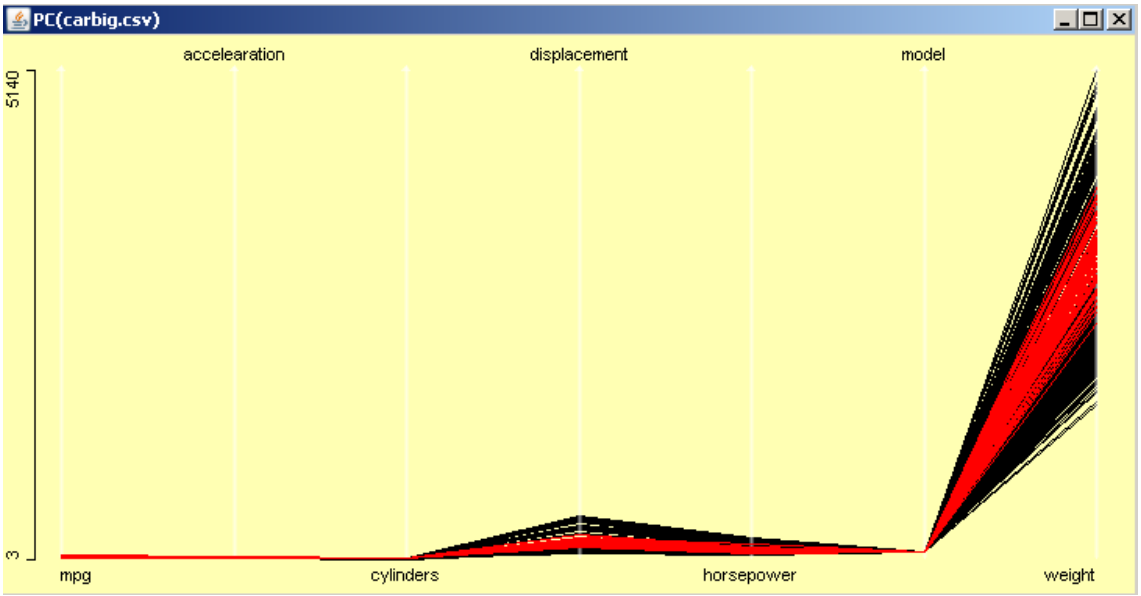


Figure 6.10. Evaluation of the drill down feature in Mondrian.

In our evaluations of GGobi and Mondrian, the dissatisfaction mainly came from the misalignment of the expectation for the interactivity provided by the visualizations as well as the result rendered. On the other hand, interactivity is also one of main problems that we try to address in parallel coordinates. The last visualization evaluated is our developed parallel coordinates with the tight integration of the HVN. For a local drill-down, we simply mouse right selection the node as indicated in the top image of Figure 6.11 and select the operation of local drill-down. The bottom image in Figure 6.11 reflects the rendered result immediately. Our drill-down functions facilitated by the HVN provide the best form of interactivity that would otherwise have been impossible to achieve in both GGobi and Mondrain.

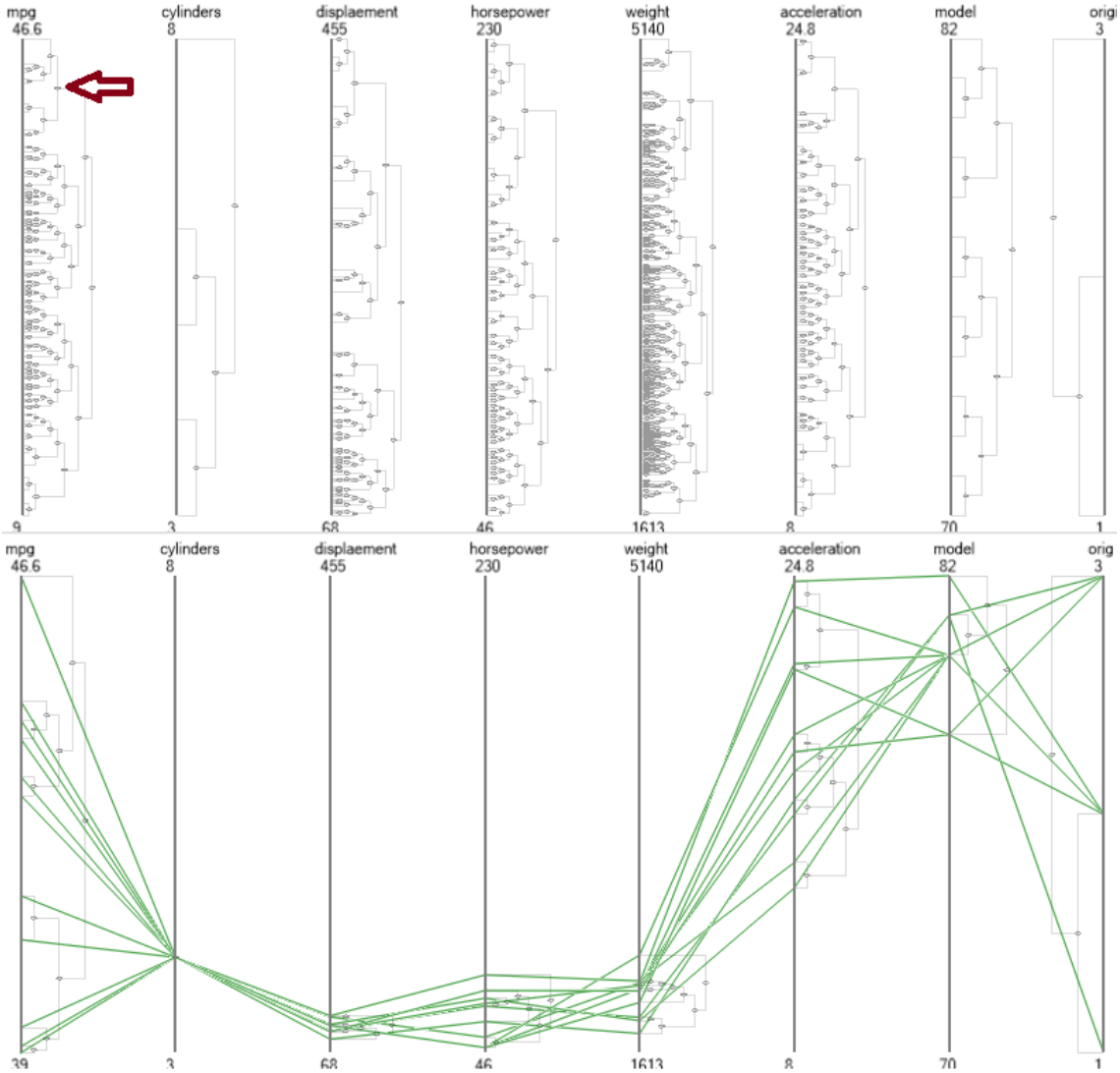


Figure 6.11. Evaluation of the local drill down feature in our HVN.

Table 6.4 lists the summary of the drill-down (or similar) feature evaluated in GGobi, Mondrian and our HVN where our approach incurs the lowest interaction cost. Obviously, we only need a simple mouse click to achieve the task rather than the input of several values via the widgets.

	Local Drill Down	Global Drill Down	Model	Interaction Cost
HVN	X	X	Point selection	Low
GGobi	X		Widget input	Median
Mondrian		X	Widget input	Median

Table 6.4. A summary of the drill down function evaluated in GGobi, Mondrian and our HVN.

Chapter 7 Case Studies

This chapter will present three case studies to demonstrate the effectiveness of the techniques described earlier. Case studies 1 and 2 are concerned with the local and global drill-down respectively. Case study 3 presents the use of the divide-and-conquer model (See Section 5.2.4) for tackling the visual analytics of a big dataset with more than 100 variables.

7.1 Case Study 1

The dataset used in this case study is Portuguese "Vinho Verde" wine data [99] which contains approximately 58776 observations (12×4898) on 12 variables namely, *fixed acidity*, *volatile acidity*, *citric acid*, *residual sugar*, *chlorides*, *free sulfur dioxide*, *total sulfur dioxide*, *density*, *pH*, *sulphates*, *alcohol* and *quality*. Nowadays, dealing with a dataset of such scale is not considered to be large but one can see the overview is already heavily cluttered in Figure 7.1. Interestingly, the nature of the data is not distributed uniformly and hence it is easy to identify a pattern for the first 8 variables as their values are mostly aggregated to the lower range.

In this case study, wine quality is the most interested variable which serves as a *decision* variable with respect to others. Owing to this, we performed a local drill-down by clicking on the data node with the highest value for *quality* variable as indicated by an arrow in Figure 7.1.

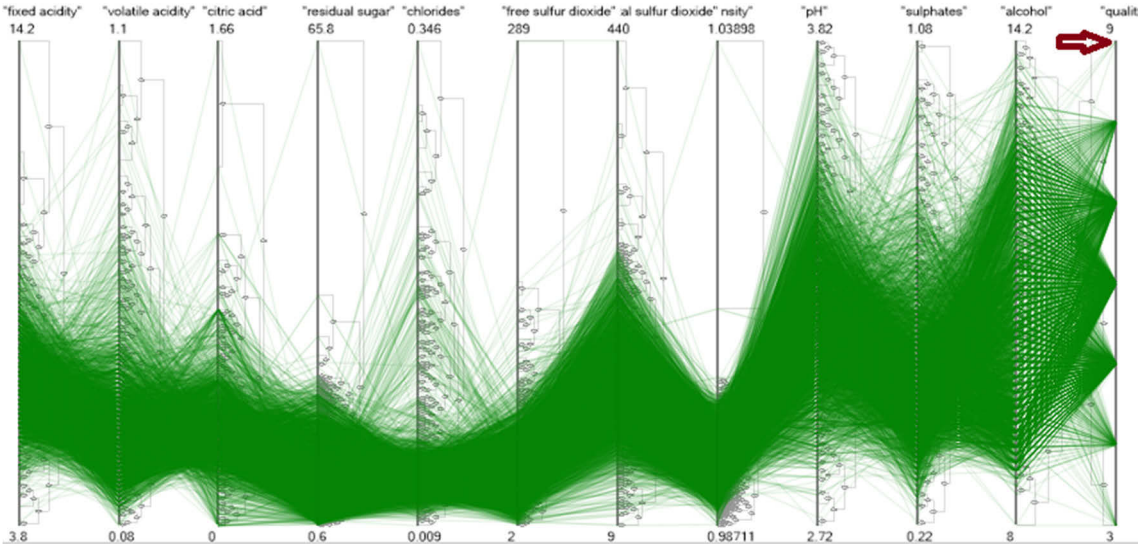


Figure 7.1. Local drill-down scenario 1.

The resultant view after a local drill-down from step 1 is presented in Figure 7.2 below. The selection set only includes those data with value 9 for *quality* (rightmost variable) value 9. One may question why the overall viewing context did not change, as we mentioned earlier the maximal and minimal range will be set to the same value when a data node is selected (see Algorithm 5.2) because the data node sits in the bottom of the hierarchy so it is a scalar value. That is, if one carefully looks at the maximum and minimum value labels for the rightmost variable in Figure 7.2 then it is evident that the value is 9 for both.

According to the result, there is a pattern for wines of the highest quality within the dataset. We could also visually identify an outlier sample in alcohol as indicated by a blue arrow because its value differs from the rest but its quality seems to be not affected due to such a difference. At this stage, we could conclude that the attributes *fixed acidity*, *volatile acidity*, *citric acid*, *residual sugar*, *chlorides*, *free sulfur dioxide*, *total sulfur dioxide* and *density* shall be lower values in order to produce the highest quality wines. Next, *pH* has attracted our attention so we decided to drill down on it locally again starting from the visual node indicated by a red arrow in Figure 7.2.

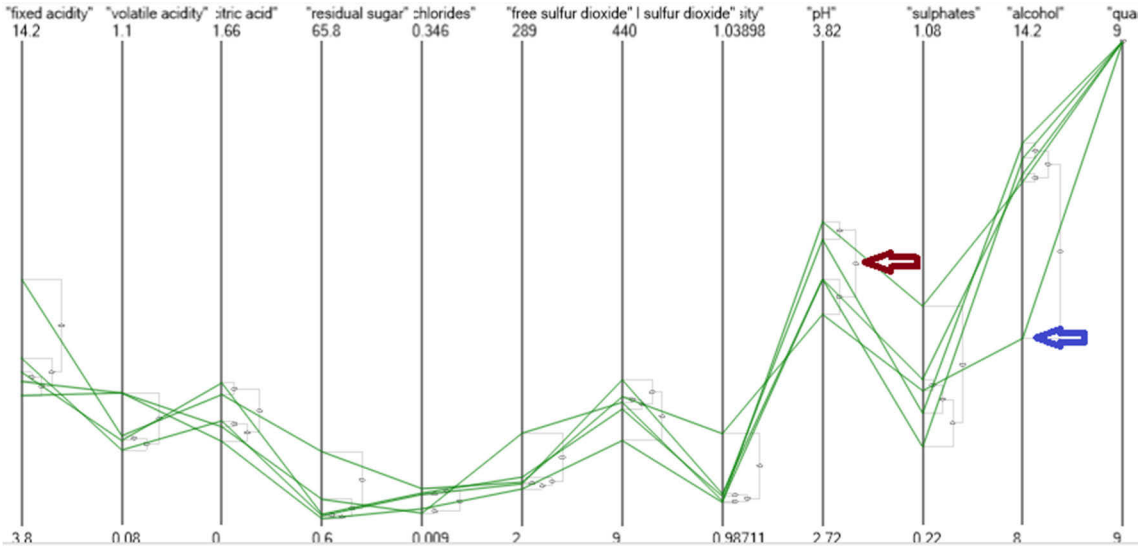


Figure 7.2. Local drill-down scenario 2.

Figure 7.3 shows the result after two consecutive local drill-down operations. One can see that the data points for *pH* have changed but overall the view remains fixed except for its adjacent variables. The numerical span of *pH* is between 2.72 ~ 3.82 and data with highest quality fall within the range of 3.2 ~ 3.41. One can also easily perceive 3 groups in *pH* through the visual inspection.

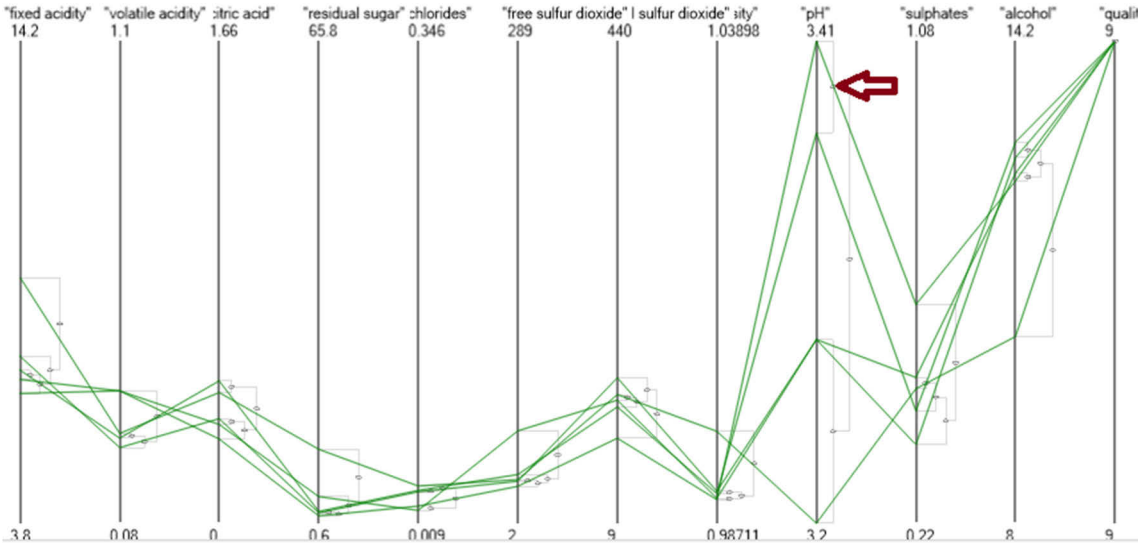


Figure 7.3. Local drill-down scenario 3.

In order to segregate data patterns, we applied brushing on a virtual node as indicated by an arrow in Figure 7.4. The result suggests that *total sulfur dioxide* tends to be lower with a relatively higher *pH* value amongst the highest quality wines.

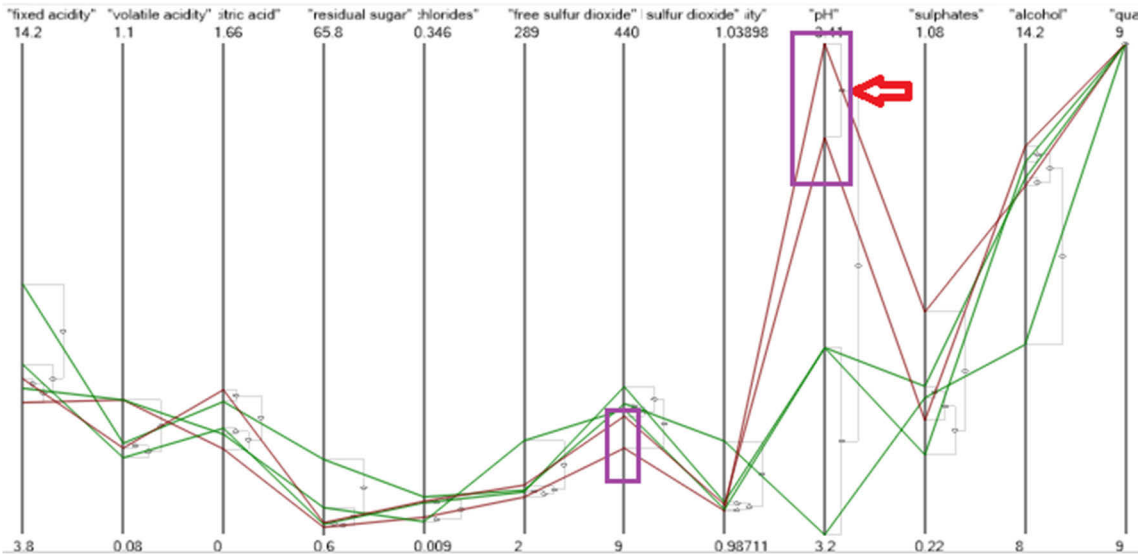


Figure 7.4. Local drill-down scenario 4.

Finally, we performed a highlighting on a polyline by hovering the mouse cursor over the data node with label pops on as shown in Figure 7.5. We concluded that wines of higher quality tended to have relatively higher *alcohol* but if it had lower *alcohol* and *pH* then the *fixed acidity* value needed to be higher to compensate.

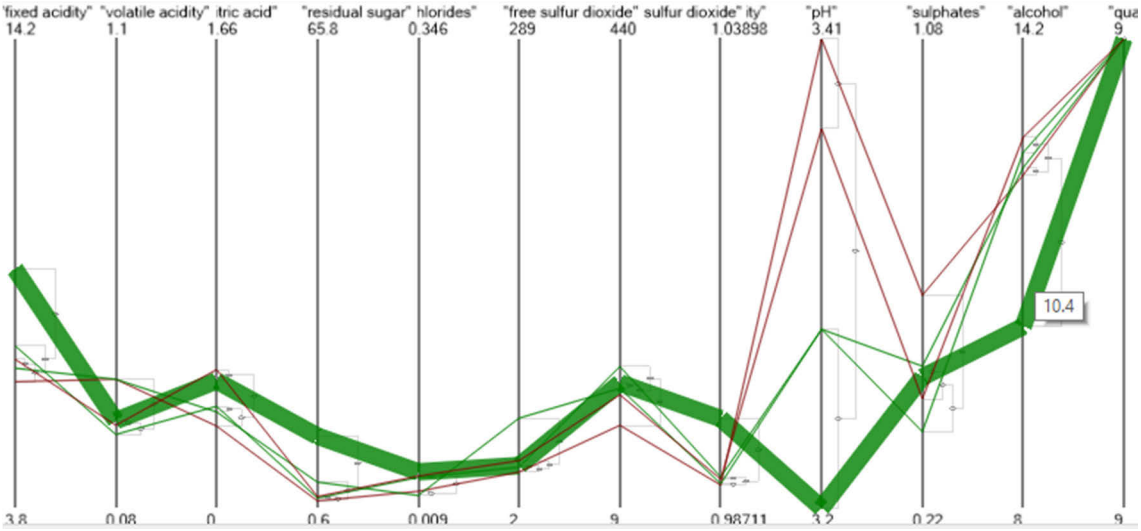


Figure 7.5. Local drill-down scenario 5.

This case study demonstrated the use of the local drill-down to arbitrarily explore the data subset interactively. In other parallel coordinates visualizations, a similar task required the user to define a numerical range which involved quantization so it was not trivial, especially as it did not contain visual hints such as data density or groups. In our parallel coordinates system, the embedded visual nodes in the display allowed for direct interaction with visual hints that would otherwise have needed multiple views or separate widgets to achieve similar functions.

7.2 Case Study 2

The dataset [100] used in this case study was collected in a Current Population Survey (CPS) 1985. It contains 534 random observations on 11 variables describing an individual's *education*, *southern residence*, *sex*, *work experience*, *union membership*, *wage level*, *age*, *race*, *occupation*, *work sector* and *marital status*.

Figure 7.6 presents a dataset overview and one can easily recognize the visualization brings out a mixture of continuous and discrete variables. Our visual analytics was first started by intuitively clicking on the virtual node (virtual data with the value 47) on *experience* with an operation of the global drill-down as indicated by a red arrow.

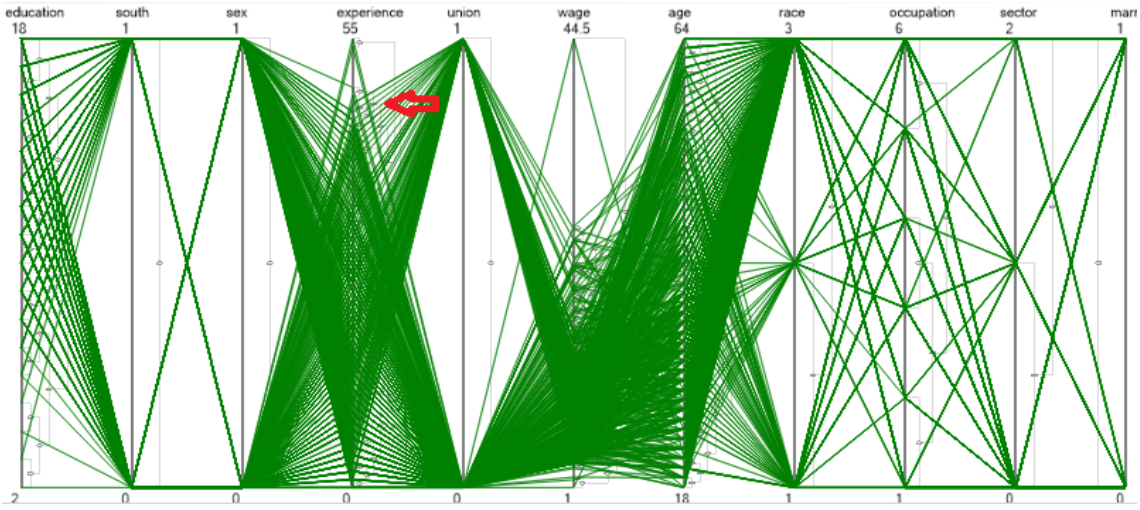


Figure 7.6. Global drill-down scenario 1.

The result presents a neat visualization as shown in Figure 7.7. According to the dataset summary, union members were paid 23% higher than non-union members and also, northern residents were paid 11% more than southern residents. With this in mind, one particular pattern attracted our interest where it could be observe that union member tended to have less wages than those without union membership. It is important to note that this statement holds true only for the selected data subset. Unfortunately, the polyline does not adequately support the visual trace of an entire path due to the nature of its geometric discontinuity at the segment junctions. There are two ways to uncover patterns under this circumstance such as brushing or highlighting. Brushing is probably not desirable for the loaded dataset where discrete variables outweigh continuous variables in quantity thus, we decided to execute a highlighting operation by hovering the mouse cursor over the virtual node, as indicated by a red arrow in Figure 7.7.

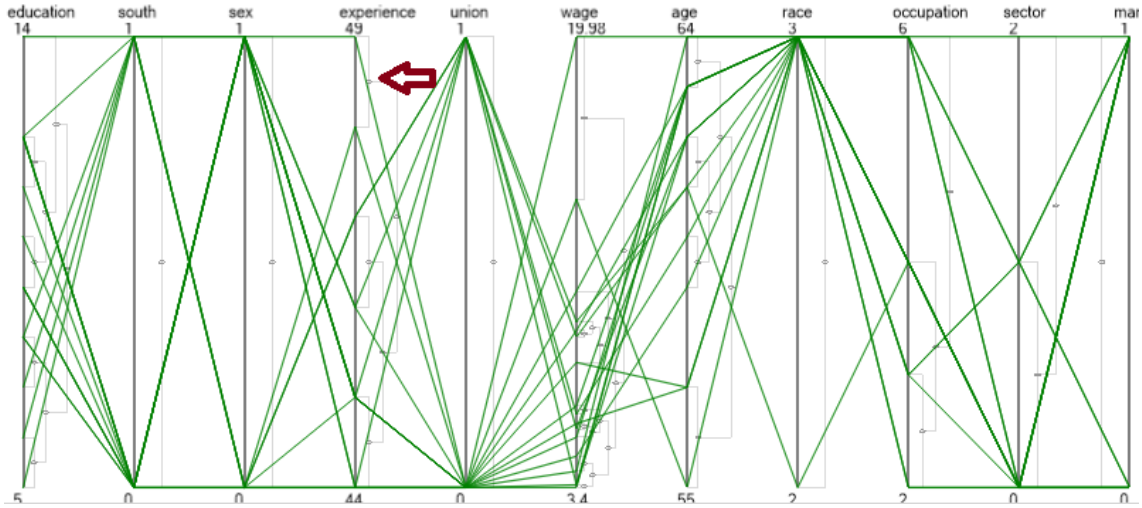


Figure 7.7. Global drill-down scenario 2.

Figure 7.8 shows the visualization of the highlighting task conducted where there were two patterns which partially overlapped near the tail. It describes observations with northern residents and non-union members with higher experience but a lesser wage. This is interesting because northern residents were paid 11% more than southern residents however, this phenomenon can be explained if we look at their education.

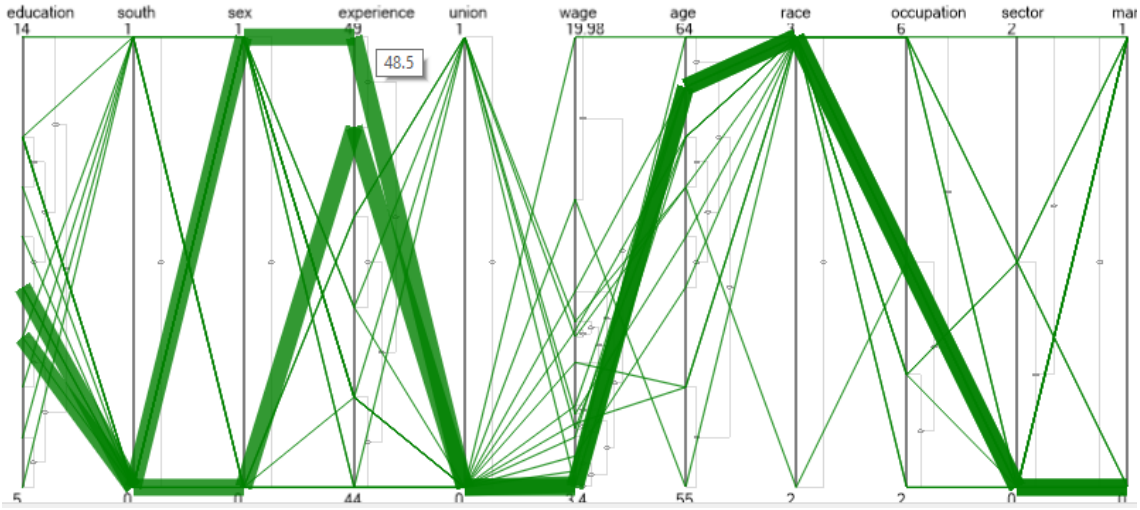


Figure 7.8. Global drill-down scenario 3.

Next, we decided to highlight observations with southern residents and the result is shown in Figure 7.9. It describes a pattern with a less experienced non-union member with a higher age and relatively higher pay albeit the residency is based on the southern area. We also noted that the residents in Figure 7.8 and Figure 7.9 were working in the same industry sector. We were able to conclude that within the selected data subset, education was the main driving factor for wage rather than sex, residency or work experience.

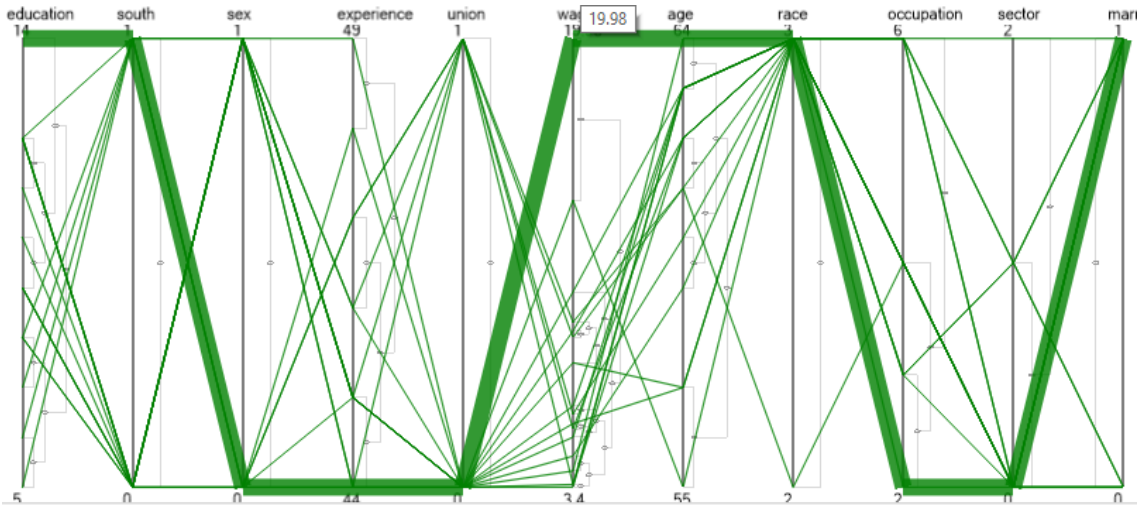


Figure 7.9. Global drill-down scenario 4.

Through these case studies, we have demonstrated the capabilities of the local and global drill-down to rapidly discover patterns in visual analytics by the assistance of the tasks defined in the dynamic selection layer (see Section 5.1).

7.3 Case Study 3

In this case study, we would like to demonstrate the exploration of a big dataset using the technique of multivariate correlation matrix described above. We have again used the National Youth Tobacco Survey 2009 (NYTS) dataset which surveyed high school youths about their attitudes, beliefs, behaviors and influences in terms of the tobacco. It contains 116 dimensions (including metadata) and 22679 data rows with approximately 2630764 data points. A visualization of the NYTS dataset in classic parallel coordinates has already been presented in Figure 5.16. The dataset with such a scale is considered extremely high dimensional and theoretically impossible for meaningful visualization due to the overplotting of the display space. Furthermore, following a thorough survey of the relevant literature, to the best of our knowledge there appear to be no case studies which used datasets of such a scale in visual analytics.

A correlation matrix is shown when the dataset has been loaded initially into the system. In Figure 7.10, one can see that there is not enough space to draw the text label but a tooltip will pop up if the user moves the mouse over a colored box which details the name of the dimensions and their coefficient of correlation. We first selected Qn33, Qn34, Qn35 and Qn37 by clicking the color boxes because they are highly correlated as hinted by the red color. The corresponding survey questions are also listed below:

Qn33: In the past 12 months, did you have to go to a stop smoking class because you were caught smoking?

Qn34: Do you think you would be able to quit smoking cigarettes now if you wanted to?

Qn35: How long can you go without smoking before you feel like you need a cigarette?

Qn37: How true is this statement for you? When I go without a smoke for a few hours, I experience craving.

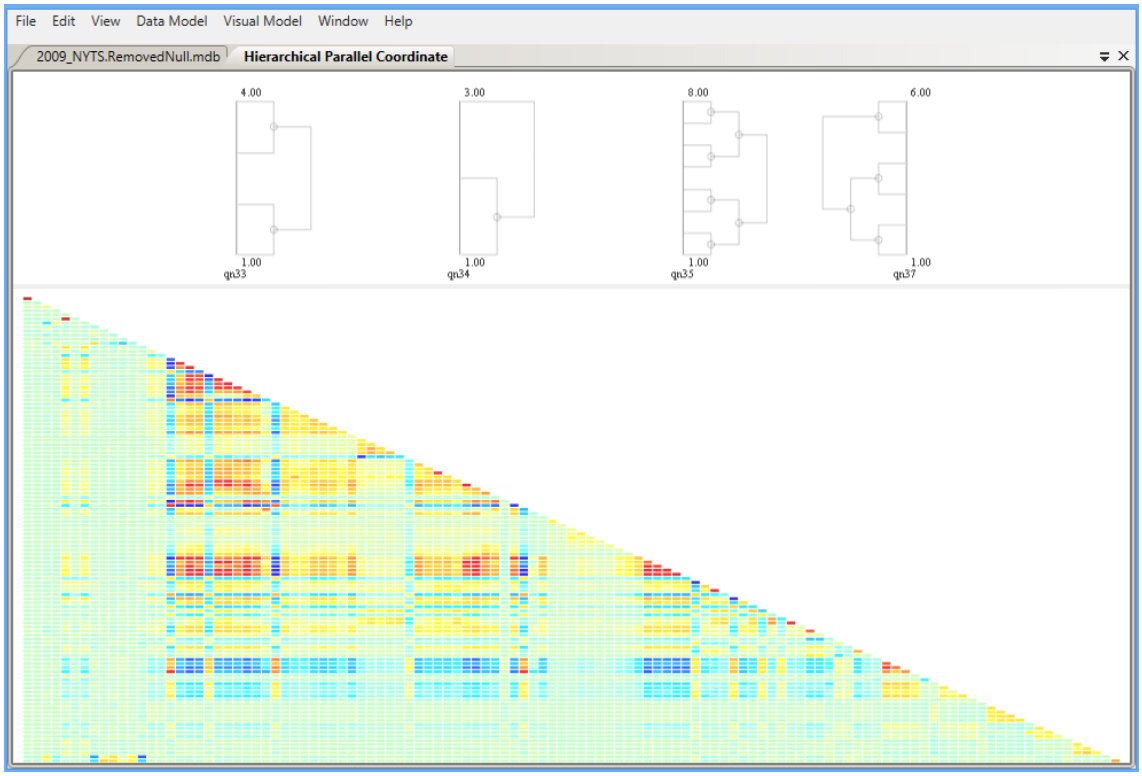


Figure 7.10. Initial view of the multivariate correlation matrix. It consists of two views where the top view is the parallel coordinates and the bottom view presents the correlation matrix.

We began the visual exploration by clicking three virtual nodes beloningg to Qn35 and each had been assigned a different color. From Figure 7.11, we found a pattern that people smoked regardless of whether they attended the stop smoking class or not and believed they could quit smoking easily if they wanted to. Also, people needing a cigarette within a 3 to 24 hour timeframe tended to agree on Qn37. Interestingly, people either answered they have quit smoking or never smoked in Qn35 even if they experienced cravings for a smoke after a few hours. This suggests that there are some false statements answered in Qn35 of the survey which further presents an area for the improvement of our system to support outlier detection visually.

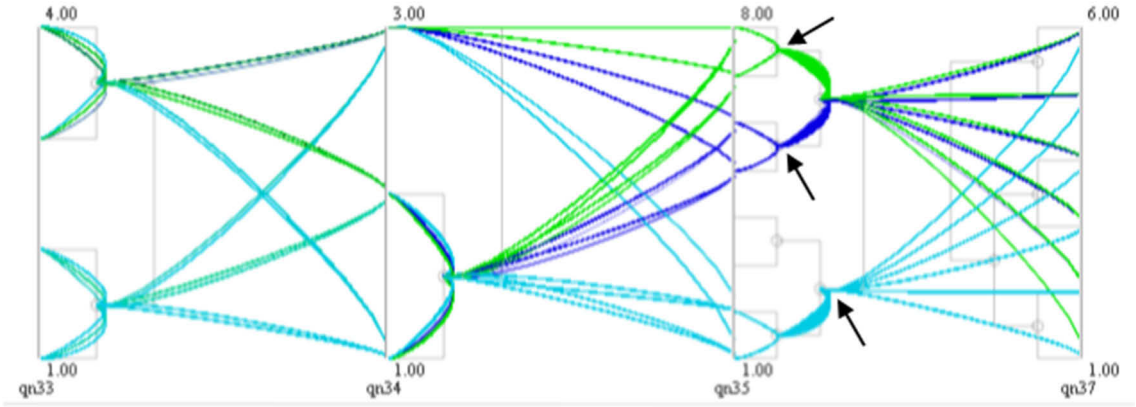


Figure 7.11. Case study step 1. The visualization result after adding Qn33, Qn34, Qn35 and Qn37 from the matrix view where the arrows indicate the mouse click over the virtual nodes.

In the next step, we clicked on a rectangle with strong correlation which added Qn58 and Qn60 to the parallel coordinates as shown in Figure 7.12 with the survey questions provided below:

Qn58: Do you think you will smoke a cigarette at any time during the next year?
Qn60: If one of your best friends offered you a cigarette, would you smoke it?

Interesting, we could visually identify a minor pattern for those who answered "*I have never smoked cigarettes*" in Qn37 and also responded that they would probably smoke at any time during the next year (Qn58) and would definitely accept a cigarette offered by one of their best friends (Qn60). It implies the risk of becoming a first-time smoker largely results from the influence of best friends. Perhaps the behavior of the friends explains the result of most first-time trials in real world cases. Unfortunately, there is no question which specifically asks about the influences leading to their decision to smoke in the first place such as friends, family, movies, TV etc. otherwise, we could derive more phenomenon in terms of the interests between the various sources.

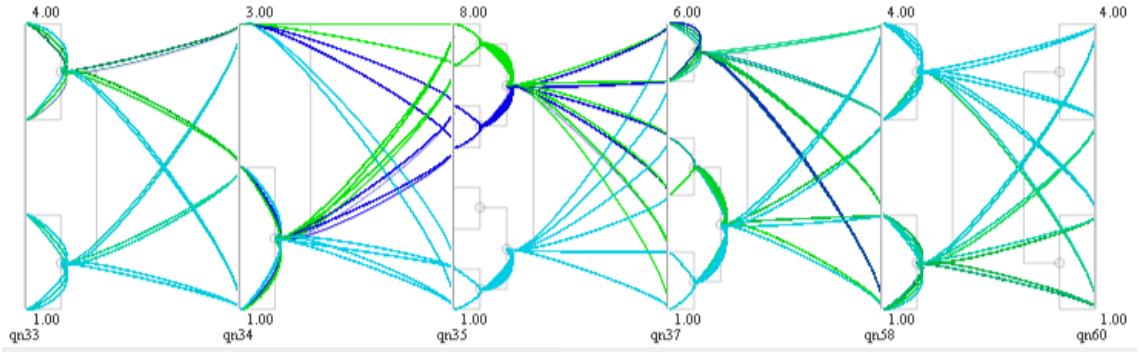


Figure 7.12. Case study step 2. The visualization result after adding Qn58 and Qn60 from the matrix view.

Next, we decided to add Qn48 and Qn49 to parallel coordinates from the matrix view with the questions provided below:

Qn48: During the past 30 days, on how many days did you smoke bidis?
Qn49: During the past 30 days, on how many days did you smoke kreteks?

Figure 7.13 has shown the visualization result of the operation where an arrow indicates a pattern where those people truly experiencing a craving for a smoke after a few hours tend to use bidis in all the past 30 days. Our interpretation is that bidi is probably attractive for truly nicotine addicted youth rather than those less addicted. Peculiarly, there was a pattern which showed that adolescents who believed that they would definitely not smoke a cigarette at any time during the next year also answered either 5 (10 ~ 19 days), 6 (20 ~ 29 days) or 7 (All 30 days) in Qn49 which means they are the frequent smokers of kretek. It is difficult for us to interpret this phenomenon because we have never ever tried both. However, a possible explanation is that the frequent smokers of kretek did not really intend to quit smoking. Perhaps, they really meant that they would not smoke a normal cigarette once they had tried kretek.

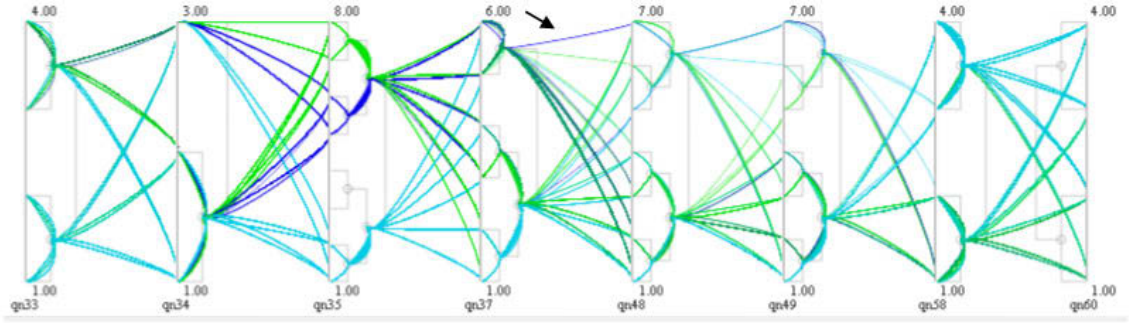


Figure 7.13. Case study step 3. The visualization result after adding Qn48 and Qn49 from the matrix view where an arrow indicates an interesting pattern.

The process can continue iteratively by adding or removing more dimensions for the dynamic view change of parallel coordinates. The purpose of this case study is to demonstrate that our framework is capable of interacting and analyzing a big dataset with high dimensionality through the guidance of the correlation matrix view. However, we do not mean to analyze them all simultaneously because human recognition cannot digest them all at once.

Chapter 8 Extended Works

In this chapter, we will introduce two extended works in relation to the interaction in multidimensional visualization.

8.1 Flow based Scatterplot Matrix

A variant of the scatterplot matrix is introduced in this section. The scatterplot is a fundamental visualization of the scatterplot matrix but it cannot explain the correlation beyond two variables. To further enhance the usability of a scatterplot matrix, we have contributed a flow based scatterplot matrix [101] for multidimensional data exploration by augmenting a scatter point to approximate its relationship with respect to a third virtual variable Z_0. Please note that, we acknowledge a similar work already contributed by Chan et al. [102][9] that was discovered prior to us, but we have further extended the idea to the application of the scatterplot matrix.

A scatter point is positioned by its data value $(X_0, Y_{0,})$ with a line attached to it. The slope indicates the positive or negative correlation with respect to (X_0, Z_0) or $(Y_{0,}Z_0)$. In global linear approximation, there is one slope so all the points reveal the same trend. Chan [102] computed the local neighborhood of radius w to smooth the local trend around a given point. In our case, we compute the local trend from the members in the class of a given point. To approximate the decision trend, the least squares in the linear regression model [103] are applied to best fit the line of a given point (X_0, Y_0) with respect to a third variable. In the linear regression model, there are two coefficients b_1 and b_0

9 We acknowledge that the idea of the flow based scatterplot was first discussed by Chan et al. [16] prior to us for studying the sensitivity, but we further extend it to the application of the scatterplot matrix and add user interaction for class exploration.

which need to be solved first, where b_1 is the slope that measures the change in Y with respect to X and b_0 is the intercept. They are defined as follow:

$$b_1 = \frac{N\sum_i^N (X_i - X_0)(Y_i - Y_0) - \sum_i^N (X_i - X_0)\sum_i^N (Y_i - Y_0)}{N\sum_i^N (X_i - X_0)^2 - (\sum_i^N (X_i - X_0))^2}$$

Equation 8.1

$$b_0 = \frac{\sum_i^N (Y_i - X_0) - b_1 \sum_i^N (X_i - X_0)}{N}$$

Equation 8.2

Where $x_i \in E(P)$ and $X_0 \in E(P)$. Substituting b_0 and b_1 into the linear equation below to interpolate the best fitting line at a point (X_0, Y_0).

$$Y_i(X_0 \pm k) = Y_0 + b_1(X_0 \pm k) + b_0$$

Equation 8.3

Where k is a desired length and please note that, we add the value of Y_0 because Y_i is a local linear approximation from a given point (X_0, Y_0).

8.1.1 Interaction by Point-to-Region

There are many cases where scatter points are partitioned into a number of classes. Given a set of points P which is further classified into the disjoint set of classes $P = \{C_1, C_2, \dots, C_n\}$ such that $C_i \cap C_j = \emptyset, i \neq j$. The point-to-region technique highlights the entire geometric region occupied upon the immediate selection of any data within the class.

Technically, the core problem of point-to-region is to build a convex hull [104] of C_i. Given two vectors $\overrightarrow{AB}$ and $\overrightarrow{AC}$, the sign of the cross product of $\overrightarrow{AB} \times \overrightarrow{AC}$ determines the direction of the triangle $\overrightarrow{ABC}$ based on the right-hand-side rule. If the cross product of $\overrightarrow{AB} \times \overrightarrow{AC}$ is positive it means the triangle is clockwise. For example, in Figure 8.1 if we attempt to find the convex hull from the clockwise direction then the negative cross product implies that $\overrightarrow{AC}$ is outmost with respect to $\overrightarrow{AB}$ otherwise, we need to swap the order of *B* and *C* which constructs a triangle. This process continues until we reach the

origin *A*. Figure 8.1 depicts the concept of using the cross product to find the convex hull on a 2D plane.

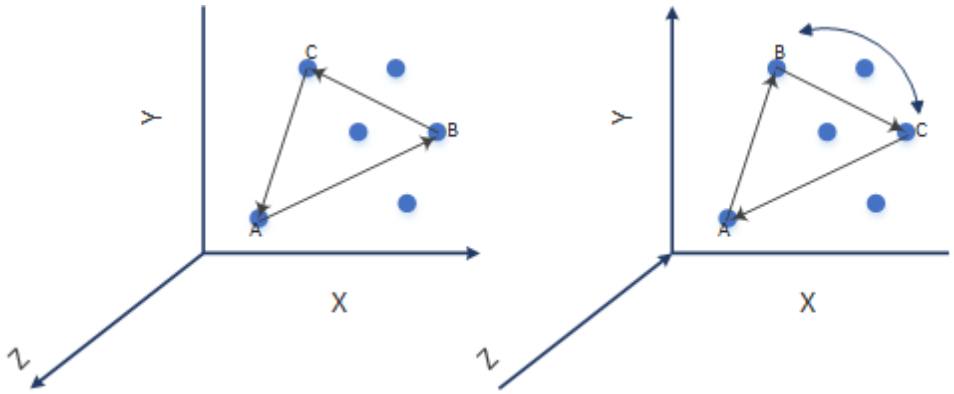


Figure 8.1 An example of the cross product. The diagram shows the application of using the cross product to find the convex hull of the data points. The direction of Z is important which determines the triangle is constructed either clockwise or counterclockwise.

The technique offers the ability of focus+context for analyzing multiple classes over the highlighted regions. An application of point-to-region has been provided in Figure 8.2. Obviously, the preliminary requirement of class data is certainly a weakness that limits its application to non-classified data points. However, its applicability can be extended by embedding the automatic data analysis of the clustering. The basic idea is to perform the K-means or hierarchical clustering on demand over the selected point in order to completely eliminate the preliminary requirement of data classification.

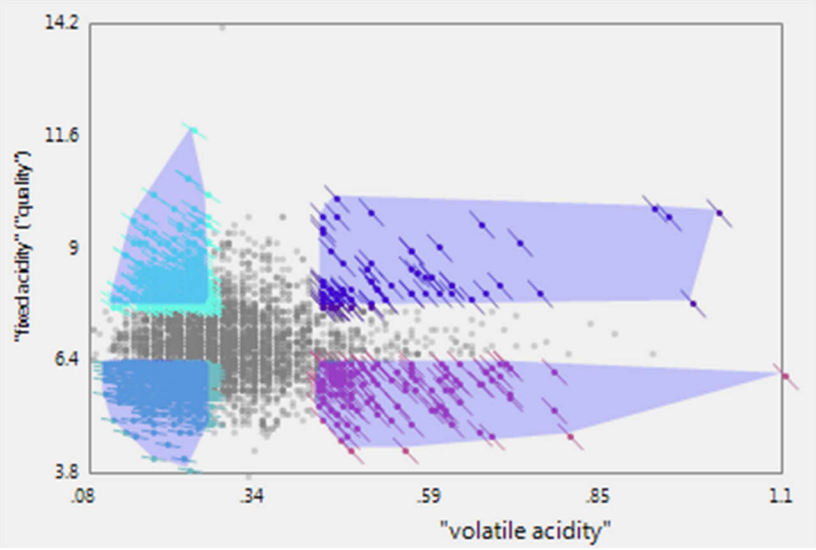


Figure 8.2 An example of point-to-region interaction. The diagram shows integrating point-to-region interaction with the flow based scatterplot matrix.

Figure 8.3 (a) and Figure 8.3 (b) provide a visual comparison between the classic and flow based scatterplot representations. Figure 8.3 (c) shows our point-to-region interaction technique. In the interaction design, we allow users to use the focus+context concept to interact with scatter points directly. This interaction method can achieve noise reduction in class selection process. For example, when visualization detects a point that has been clicked, the entire convex hull of a corresponding class is highlighted and it greys out the background of the convex hull in the meantime.

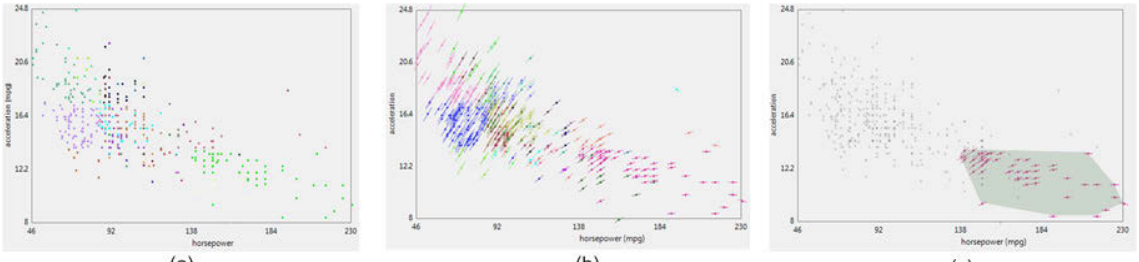


Figure 8.3 From scatterplot to flow based scatterplot. (a) A classic scatter plots visualization. (b) Adding the decision flow where plots are augmented with respect to the decision variable. (c) Interaction (mouse click) by using point-to-region concept: that is, a point click causes an entire convex hull (a class) highlighted.

An application of flow based scatterplot matrix is shown in Figure 8.4 where the upper matrix remains the traditional scatterplot and the lower matrix embeds the flow based scatterplot.

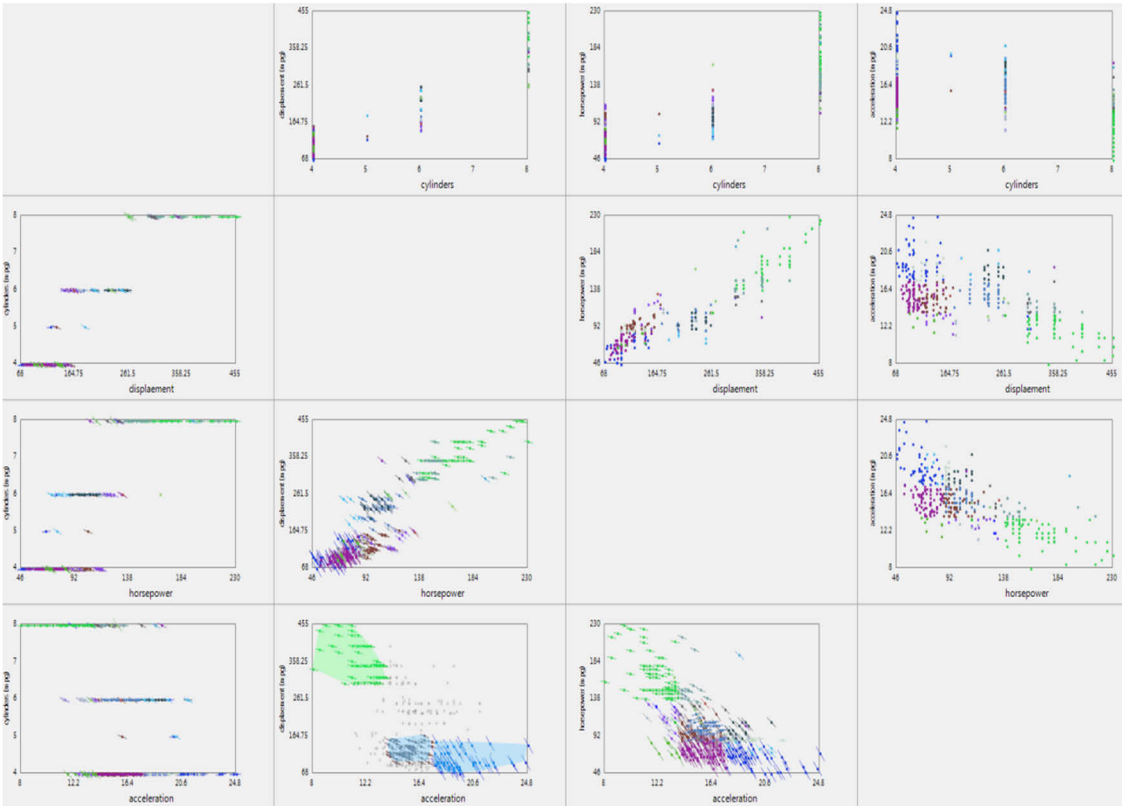


Figure 8.4 A visualization of the flow based scatterplot matrix. The car dataset is used in this example where the upper and lower triangular matrix display the classic scatterplot and flow based scatterplot respectively.

8.2 Space Filling Multidimensional Visualization

Space Filling Multidimensional Visualization (SFMDVis) is a novel technique of multidimensional visualization that is primarily designed to avoid overplot and visual clutter. According to the categories defined by Keim and Kerigel [17], it is classified as a pixel oriented technique. For a similar prior work, a pixel bar chart visualization developed by Keim [105] will be a good example.

8.2.1 Properties and Definitions

Multidimensional dataset consists of a set of variables $X = \{X_1, X_2, X_3, \dots, X_N\}, \exists! X_i \in X$ with M observations. For simplicity, we typically organized the dataset in the form of an $M \times N$ matrix as follows:

$$\mathrm{X} = \begin{pmatrix} X_{1,1} & \cdots & X_{1,N} \\ \vdots & \ddots & \vdots \\ X_{M,1} & \cdots & X_{M,N} \end{pmatrix}$$

The column vector in the matrix is expressed as $D = \{d_1, d_2, d_3, \dots, d_N\}$. P_i is used to denote an observation such that $P_i = \{d_{X_1}, d_{X_2}, \dots, d_{X_N}\}$ and $P_i \in P$. P and D describes the fundamental row and column vectors that we will be working with. Please note that, the notation of d_{X_i} will also be used individually for element indexing in P_i for convenience. In addition, we expect the drawing surface to be rectangular. ω and h denote the width and height respectively in pixels where $[\omega \times h]$ defines the drawing bound. Given a set of variables X, we further divide the ω into the equal length as

$$\omega_X = \frac{\omega}{(N-1)}$$

Where ω_X simply defines the width that we allocate to each variable in a horizontal line for color mapping and $\overline{\omega_{X_i}\omega_{X_j}}$ refers to the line segment. Please refer to Figure 8.5 for clarity.

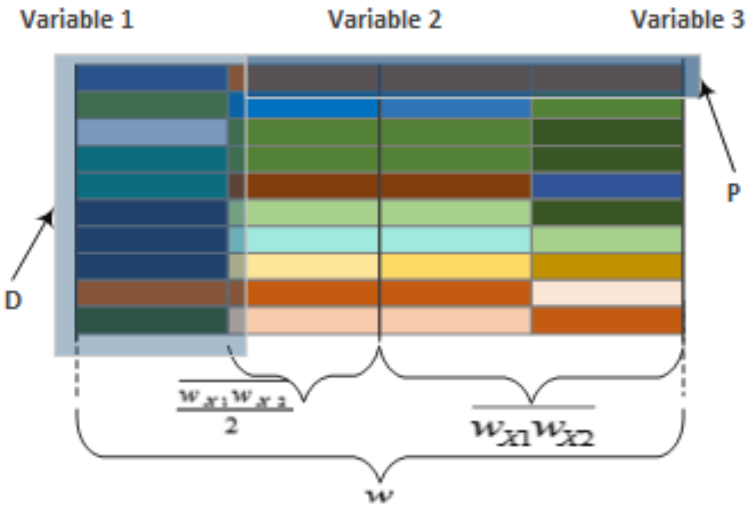


Figure 8.5. The properties of SFMDVis.

8.2.2 SFMDVis

The basic idea behind the space filling visualization is the representation of a data row P_i by a horizontal line which serves as a fundamental geometric primitive. The line is further segmented by the colors to denote the values with respect to each variable X_i. The width of P_i equals to ω and the height occupies a unit size in pixels on the screen multiplied by the zooming factor γ. For example, if the unit size is one pixel and $\gamma = 2$, then each line will occupy 2 pixels in height.

In a multidimensional dataset X, variables might scale differently. Therefore, we need to apply a normalize function in order to remove the discrepancy such that $f : P \rightarrow \hat{P}$.

$$\hat{P} = \sum_{P_i \epsilon P} \sum_{D_{X_i} \epsilon P_i} \frac{\left(D_{X_i} - X_{min}\right)}{(X_{max} - X_{min})} : D_{X_i} \in X_i$$

Equation 8.4

Where $\hat{P}_i = \left\{\widehat{D_{X_1}}, \widehat{D_{X_2}}, \dots, \widehat{D_{X_N}}\right\}$ holds the normalized values with respect to each variable X_i within the range $[0,1]^D$ and X_{min} and X_{max} denote the minimal and maximal values of a target variable such that $[X_{min}, X_{max}]$.

Recall that $\hat{P}_i$ will be mapped to a line that further consists of a set of segments $\left\{\overline{\omega_{X_1}\omega_{X_2}}, \overline{\omega_{X_2}\omega_{X_3}}, \dots, \overline{\omega_{X_{N-1}}\omega_{X_N}}\right\}$ and each segment is coded by two colors. This is because each segment holds two end points and each point represents a variable X_i. Let C denotes the color vectors and $C \Rightarrow \{R, G, B\}$ be a class in C. We can map a normalized data $\widehat{D_{X_\iota}}$ to the nearest C_i and it is given as:

$$C_i = \left\lfloor \frac{\widehat{D_{X_\iota}}}{1.0/|C|} \right\rfloor \mapsto C$$

Equation 8.5

Where $\lfloor * \rfloor$ denotes a floor function. Recall that, the maximal normalized value is 1 so we divide it by the number of colors to work out the normalized cut point. We then divide the normalized data by the cut point to index a color. Therefore, the color order is important since the color progression is often perceived as a value magnitude with respect

to a variable. Similarly, we can work out the cut point range that C_i holds in data value (non normalized) with respect to X_i by the following equation:

$$k = \frac{(X_{max} - X_{min})}{|C|}, \left[X_{min} + (i \times k), X_{min} + \big((i+1) \times k\big)\right]$$

Equation 8.6

Where i and k refer to the bin index and cut point respectively. Recall that, we have mentioned earlier that each end point of a segment represents a variable so it is further divided by 2 as $\overline{\omega_{X_1}\omega_{X_2}}/2$ for painting the value of each variable. Thus, given the tuple $\langle\overline{\omega_{X_i}\omega_{X_j}}, C_i, C_j\rangle$, there are two *DrawLine* calls required to paint the sub-segments with length $\left[\omega_{X_i}, \overline{\omega_i\omega_{X_j}}/2\right]$ and $\left[\overline{\omega_i\omega_{X_j}}/2\,, \omega_j\right]$ for C_i and C_j respectively.

The core algorithm of SFMDVis has been completely described in Algorithm 8.1 where we pass two arguments *P* and *X* that hold the row and column vectors respectively.

```
1.  procedure RenderSpaceFilling(P, X)
2.      curPixel ← 0
3.      /* Draw the vertical coordinates. */
4.      for i := 0 to |X| do
5.          DrawLine((i × ω_X), 0, (i × ω_X), h, Gray)
6.      end
7.      /* Iterate through each data row. */
8.      for P_i ∈ P do
9.          P̂_i = normalize(P_i)
10.         if IsDataRowSelected(P̂_i) then
11.             /* Draw the line segment. */
12.             for j := 0 to |X| − 1 do
13.                 C_j ← mapColor(D̂_{X_j})
14.                 C_{j+1} ← mapColor(D̂_{X_{j+1}})
15.                 mid ← (ω_{X_j} + ω_{X_{j+1}})/2
16.                 /* Apply the zooming factor. */
```

17. **for** $k := 0\ to\ k < \gamma$ **do**
18. $l \leftarrow (curPixel + k)$
19. $DrawLine\left(\omega_{X_j}, l, mid, l, C_j\right)$
20. $DrawLine\left(mid, l, \omega_{X_{j+1}}, l, C_{j+1}\right)$
21. **end**
22. **end**
23. **end**
24. $curPixel \leftarrow curPixel + \gamma$
25. **end**
26. **end procedure**

Algorithm 8.1. The core algorithm of SFMDVis.

Figure 8.6 illustrates a visualization of SFMDVis where one can see that there is no visual clutter and overplotting in SFMDVis because data items do not overlapped to each other. There two problems are commonly seen in parallel coordinates due to its spatial arrangement of data items. Also, every data item is directly selectable that makes SFMDVis really distinctive from others.

Overall, the interactive techniques developed within SFMDVis have influenced our framework significantly. For example, the zooming technique developed in SFMDVis has been extended to interactive drill-down (Section 5.2.1 and 5.2.2) in parallel coordinates.

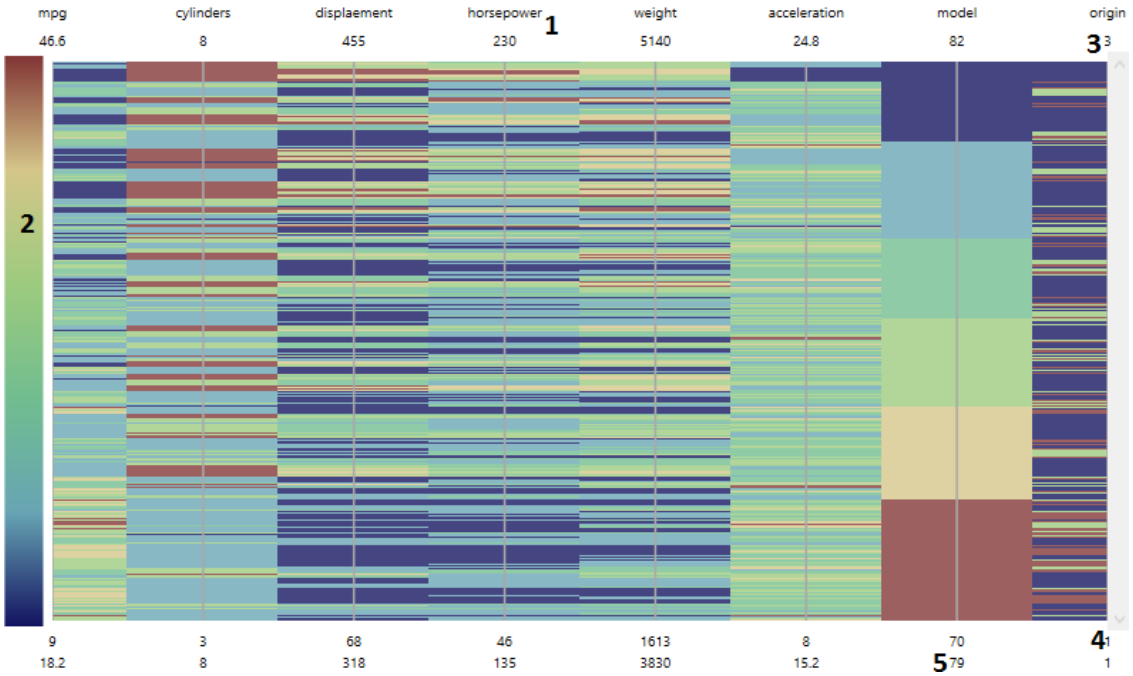


Figure 8.6. A visualization of SFMDVis. (1) Text labels that describe the variable names. (2) Color legend. (3) and (4) denote the maximal and minimal value range. (5) Dynamic values and these refer to the data row pointed to by the mouse cursor.

8.2.3 Color Models

This section describes two supported color schemes in SFMDVis namely, the RGB and single-hue.

8.2.3.1 RGB Color Ramping

In the RGB color ramping scheme, red, green and blue are commonly chosen to express the higher, middle and lower magnitude of a value. This is probably because people naturally tend to associate red and blue with hot and cold respectively. In RGB ramping, the number of color bins are determined by a *ramping factor* denoted as r which also controls the variation of the colors when progressing in between blue $(0,0,1)$ to red $(1,0,0)$. Figure 8.7 has demonstrated a color legend of RGB ramping with $r = 6$.

Figure 8.7. An illustration of using RGB color remapping to denote the value magnitude. The red, blue and green denote the highest, lowest and middle value magnitude.

The higher ramping factor separates colors in smaller step changes that may possibly affect the ability of the human eye to interpret the magnitude of the value if the change is really subtle. For example, if r is too small then the number of color bins might be insufficient to represent the data distribution. On the other hand, the usability will decline rapidly when r increases because the human eye might not be able to discern the subtle change in adjacent colors. When determining the proper ramping factor, we have noticed a study contributed by Healey [106] for choosing the effective colors in data visualization. According to the study, the result has suggested the human visual system can quickly identify up to 5 classes of color in parallel but the response time of target identification has increased during 7 and 9 color studies. Based on this observation, the ramping factor is set to 6 by default in our visualization for RGB color ramping.

The algorithm implemented in our visualization is based on the implementation described by Bourke [107]. Although, we have modified the original work slightly but in general, the principal is the same so it will not be reproduced here.

8.2.3.2 Single-Hue Ramping

With multiple classes of color in RGB ramping, the user may need to reference the color legend frequently if they cannot translate the spectrum to the magnitude that it implies intuitively. For this reason, the single-hue is added as an alternative ramping scheme that aims to support users who are not comfortable with the RGB style colors. Please note that we offer the interaction possibilities for the user to switch between the color schemes by right-clicking the mouse over the color legend which will then display the color scheme options in a menu widget.

Single-hue is commonly applied in the choropleth map for mapping the magnitude of the data (often aggregated) with respect to a geographic location. In single-hue ramping, the appearance of the color is progressed from dark to a light shade of the same color but with different levels of saturation and lightness, and therefore, the method is named single-hue. The basic idea of single-hue is illustrated in Figure 8.8 by Hue,

Saturation and Lightness (HSL) where the hue value ranges from $[0,255]$ and, saturation and lightness are both measured in percentage between $[0,100]^{\%}$.

Figure 8.8. An example of single-hue progression in the purple color. The corresponding HSL values are (270, 100%, 25%), (266, 57%, 36%), (243, 31%, 61%), (245, 29%, 69%), expressed from left to right.

Our color selection is based on ColorBrewer [108] which is an excellent online tool that provides prebuilt-in colors for sequential, diverging and qualitative schemes. In the visualization, we provide the single-hue color options of blue, green, orange, purple and grey scale for user preference. Figure 8.9 has shown the visualization results of using the blue and green single-hue.

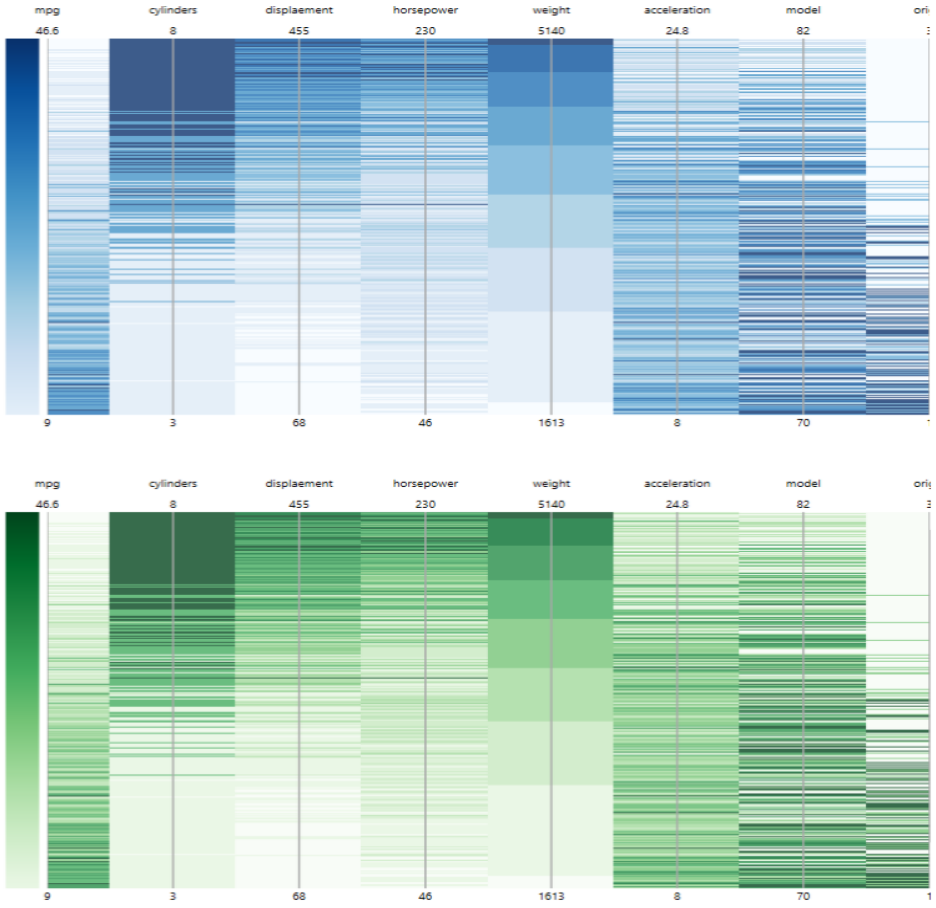


Figure 8.9. Single-hue color ramping in blue and green colors.

8.2.4 Interaction Techniques in SFMDVis

In this section, we will introduce the interactive techniques supported in SFMDVis.

8.2.4.1 Zooming

In SFMDVis, the zooming enlarges a pixel size for scaling a pattern. Sometimes it can be difficult to notice a weak pattern through the visual inspection since an entire pattern might occupy just few pixels in height. To address this issue, we have incorporated a zooming technique which can be activated by pressing the Ctrl-key and scrolling the mouse wheel in the meantime to control the zooming factor γ. The zooming factor is in the range of $[1, 10]$. Technically, γ can be infinitely large $[1, \infty]$ but we believe that a maximal value of 10 is enough in most cases. For example, if the unit size is one pixel with $\gamma = 10$ then each line will occupy 10 pixels in height which shall be large enough to perceive a pattern. Figure 8.10 compares two visualizations with the zooming factor set to 1 and 10. The color pattern for every single line can be observable easily when zooming factor has set to 10 where each line is 10 times larger than its default size and that is the maximal value supported.

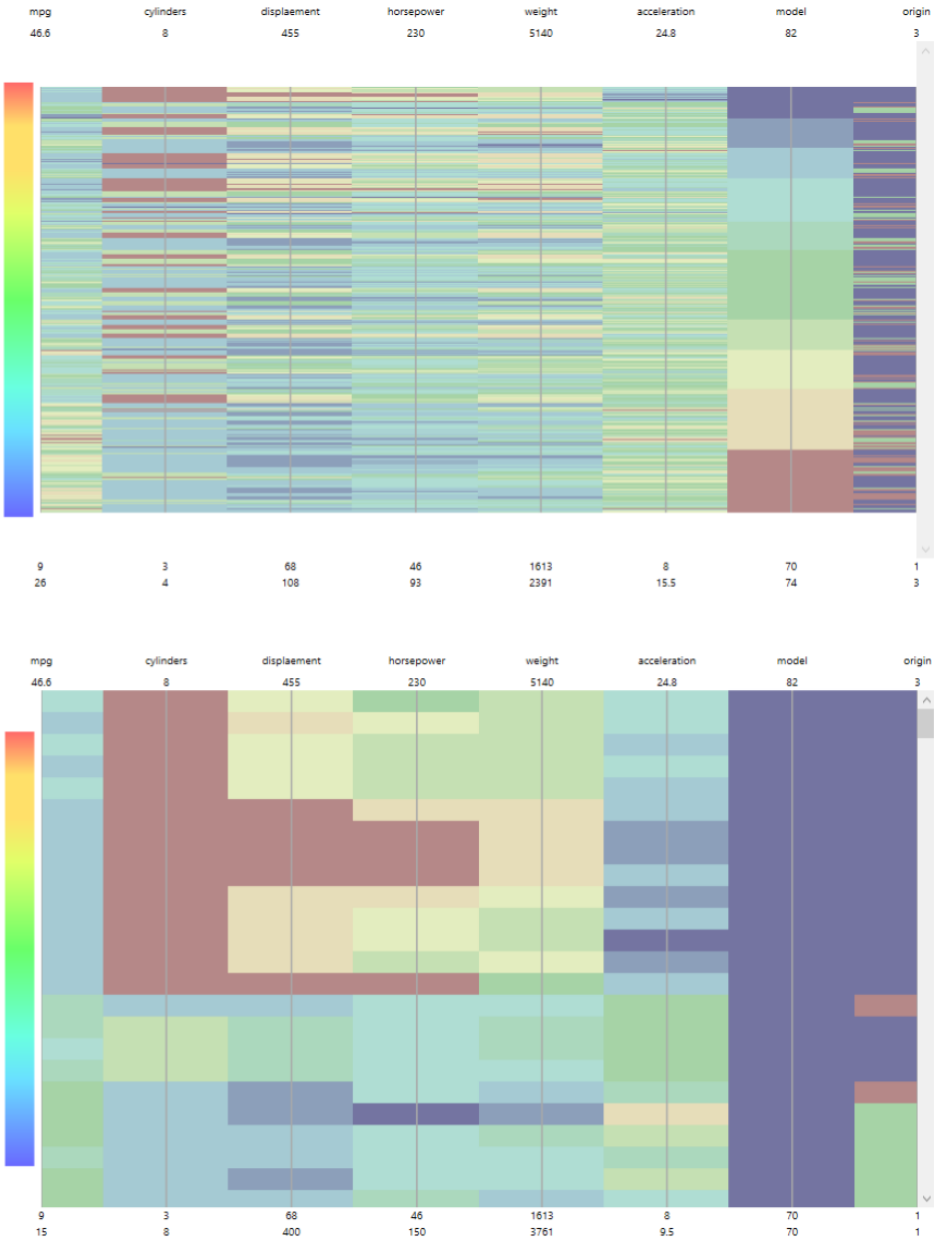


Figure 8.10. Zooming in SFMDVis. (Left) The overview of the car dataset with $\gamma = 1$ which is the default and that means, there is no scaling at all. (Right) The car dataset with zooming factor $\gamma = 10$.

8.2.4.2 AND and OR Operator for Data Selection

In SFMDVis, the technique to interact with data is a point-to-color region. The main consideration that we do not filter data based on an absolute value is to maintain visual consistency and expectation. For example, when a mouse clicks on a point with a color C_i for a variable X_i, the user intuitively expects that for these lines P_i with color brushing

$C_{D_{X_i}} \neq C_i$ to be filtered out. Recall that, we deal with data in classes rather than absolute value so if we strictly filter data based on their values rather than their classes then the visual consistency cannot be maintained. When a mouse click within a drawing region $[\omega \times h]$ is detected, we pass the x-coordinate to the following equation to determine the matrix column index $n \mapsto X$ such that $0 \leq n < |X|$.

$$f(x) = \begin{cases} X_i, & \sum_{X_i \in X} \omega_{X_i} \leq x \leq \overline{\omega_{X_i}, \omega_{X_{i+1}}}/2 \\ null, & otherwise \end{cases}$$

Equation 8.7

Where $(i + 1) < |X|$ to ensure that we access an element within the vector bound. For finding the matrix row index $m \mapsto X$ such that $0 \leq m < |P|$, we need to divide the y-coordinate by the zooming factor γ that is written as:

$$f(y) = {}^{y}/_{\gamma} + {}^{h_{offset}}/_{\gamma}$$

Equation 8.8

Where h_{offset} denotes the view offset in screen coordinate to the original due to the scrolling effect. These information are persisted by a tuple $\langle m, n \rangle$. In SFMDVis, the user is able to select data with the AND and OR operator in order to control the visibility of the interested data for comparison with greater flexibility. Figure 8.11 illustrates the operation of data selection using the AND and OR operator.

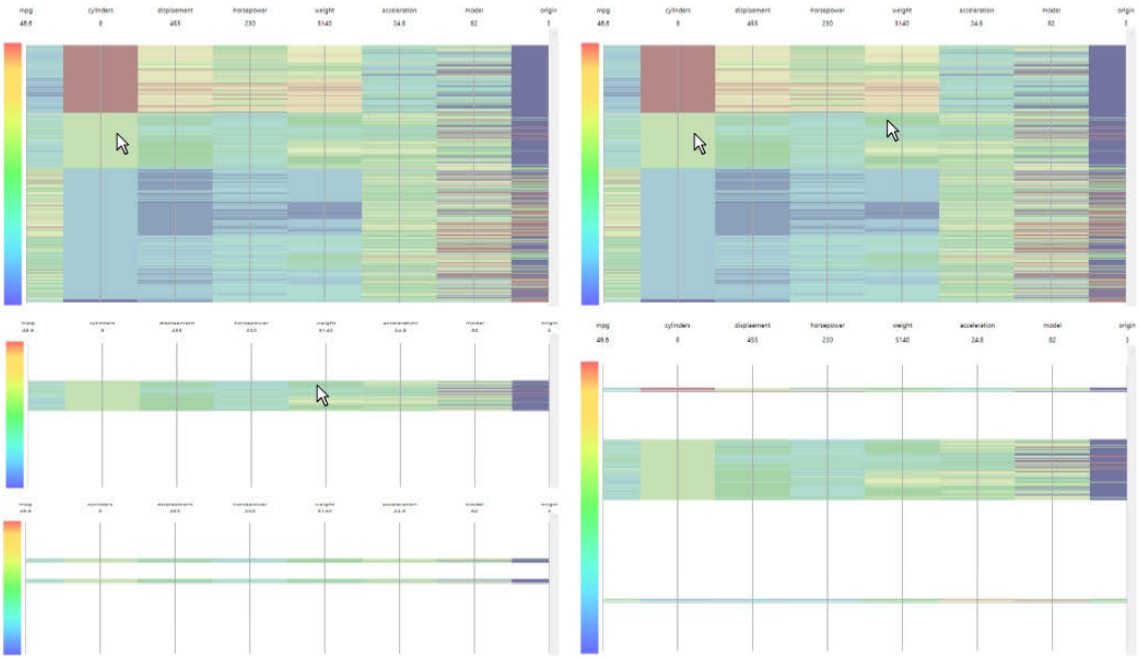


Figure 8.11. Interactive AND and OR data selection in SFMDVis. (Left) Data selection with AND operator. (Right) Data selection with OR operator. The cursor indicates the mouse clicks. The AND operator is useful in filtering out the data while OR operator can be used to find the data pattern between groups rapidly.

8.3 Discussion

In this chapter, we have first introduced a flow-based scatterplot matrix with a point-to-region technique for interacting with a class of data. In addition, we have further introduced a novel multidimensional visualization called SFMDVis which does not use traditional coordinated system as well as classic geometric primitive to represent multidimensional data. Those two works are served as the extended contributions to the interactive mechanism in multidimensional visualization in addition to the virtual node.

Chapter 9 Conclusion

9.1 Summary

In summary, this dissertation presented models and techniques for interactive visual analytics in multidimensional visualization particularly in parallel coordinates. Overall, the materialized contributions are summarized as below.

Chapter 3 introduced a new framework of visual interactions by refining Yi's 7-layer models. Existing frameworks tend to classify interactive tasks in a fine-grained manner based on the nature of the operations or the user's intent. We argue that this is not necessary and the interactions can be narrowed down to inputs and output if one models visualization as a function. Therefore, we propose a 3-layer framework based on Yi's model. The new model broadly classifies interactive visual analytics into 3 categories as data selection, visual techniques for view change, and data analytics techniques for reasoning. Formally, the layers in the new model are dynamic selection, dynamic viewing and dynamic scoping of data. The advantage of this new model makes it easy to understand and allows for better interpretation of the layered structure of visual interactions.

In Chapter 4, a novel and sophisticated technique of data selection has been contributed, called the hierarchical virtual node (HVN). The chapter also provides comprehensive technical and implantation details. The basic idea is to interpolate visual nodes in parallel coordinate hierarchically for data selection. To the best of our knowledge, it is the first technique that enables users to interact directly with data in parallel coordinates using a point-selection (mouse click) method. There are many advantages of HVN. For example, point selection is always more intuitive, efficient and accurate than other methods. Also, it enables a multi-level of data interaction through the hierarchical grouping of the data. Another advantage that was not realized before the implementation is that it enables users to perceive the data distribution through the distribution of the virtual nodes and such information is often lost due to overplot in parallel coordinates.

Chapter 5 presented the HVN-oriented interactive tasks for visual analytics based on our model introduced in Chapter 3. These tasks were carried out in the system developed.

9.2 Final Conclusion

In conclusion, we have taken the research challenges and satisfied the goals defined. The proposed HVN which is the core technique of interaction has solved the issue to interact with multidimensional data directly in parallel coordinates. It opens the applications of many analytic tasks introduced earlier but is not limited to these. Moreover, they can be easily carried out by a point-selection technique which is the most intuitive model for human interaction and this would have otherwise been impossible to achieve by other techniques.

In future work, the concept of the virtual nodes will be extended to other visualizations where applicable. The theoretical development of virtual interpolation will also continue. For example, the interpolation of the virtual node is based on the basis of hierarchical clustering but it is also possible to interpolate nodes based on the density for interaction though, this needs further study to prove its feasibility.

Appendix A Publications

- T. H. Huang, M. L. Huang, Q. V. Nguyen, L. Zhao, Space-Filling Multidimensional Visualization (SFMDVis) for Exploratory Data Analysis, In Proc. of the 7th Inter. Sym. On Visual Information Communication and Interaction, pp. 19-28, 2014.

- T. H. Huang, M. L. Huang, K. Zhang: An Interactive Scatter Plot Metrics Visualization for Decision Trend Analysis. ICMLA (2), pp. 258-264, 2012.

- T. H. Huang, M. L. Huang, Jesse S. Jin: Parallel Rough Set: Dimensionality Reduction and Feature Discovery of Multi-Dimensional Data in Visualization. ICONIP (2), pp. 99-108, 2011.

- M. L. Huang, T. H. Huang, J. Zhang: TreemapBar: Visualizing Additional Dimensions of Data in Bar Chart. IEEE Intel. Conf. on Information Visualization, pp. 98-103, 2009.

- T. H. Huang, M. L. Huang: Visualization of Individual's Knowledge by Analyzing the Citation Networks. CGIV 2007, pp. 465-470, 2007.

- T. H. Huang, M. L. Huang: Analysis and Visualization of Co-authorship Networks for Understanding Academic Collaboration and Knowledge Domain of Individual Researchers. CGIV 2006, pp. 18-23, 2006.

Bibliography

[1] W. J. Frawley, G. Piatetsky-Shapiro and C. J. Matheus, "Knowledge discovery in databases: An overview," *AI Magazine (AAAI),* vol. 13, no. 3, pp. 57-70, 1992.

[2] P. C. Wong and R. D. Bergeron, "30 Years of multidimensional multivariate visualization," in *Scientific Visualization*, p. 3-33, 1997.

[3] G. E. Moore, "Progress in digital integrated electronics," in *IEEE International Electron Devices Meeting*, pp. 11-13, 1975.

[4] A. Unwin, M. Theus and H. Hofmann, Graphics of large datasets: visualizing a million, Springer, 2006.

[5] R. Kosara, "Visualization criticism – The missing link between information visualization and art," in *In Proc. of 11th Conference on Information Visualization IV'07*, p. 631-636, 2007.

[6] J. Meyer, J. Thomas, S. Diehl, B. Fisher, D. Keim, D. Laidlaw, S. Miksch, K. Mueller, W. Ribarsky, B. Preim and A. Ynnerman, "From visualization to visually enabled reasoning," in *Scientific Visualization: Advanced Concepts*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010, p. 227–245.

[7] E. Kleiberg, H. van de Wetering and J. J. van Wijk, "Botanical visualization of huge hierarchies.," in *In Proc. of IEEE Symp. on Information*, pp. 87-94, 2001.

[8] J. J. van Wijk, "The value of visualization," in *IEEE Visualization*, pp. 79-86, 2005.

[9] D. Keim, G. Andrienko, J. D. Fekete, C. Gorg, J. Kohlhammer and G. Melancon, "Visual analytics: definition, process, and challenges," in *Information Visualization in volume 4950 of LNCS*, Berlin, Springer, 2008, pp. 154-175.

[10] D. A. Keim, "Information visualization and visual data mining," *IEEE Tran. Visualization and Computer Graphics,* vol. 8, no. 1, pp. 1-8, 2002.

[11] D. A. Keim, F. Mansmann, J. Schneidewind and H. Ziegler, Visual analytics: Scope and challenges. Visual Data Mining, Springer, Lecture Notes In Computer Science (LNCS), 2008.

[12] J. J. Thomas and K. A. Cook, Illuminating the Path: The Research and Development Agenda for Visual Analytics, IEEE Press, 2005.

[13] A. Inselberg, "The plane with parallel coordinates," *The Visual Computer,* vol. 1, no. 2, pp. 69-91, 1985.

[14] J. S. Yi, Y. A. Kang, J. Stasko and J. Jacko, "Toward a deepe understanding of the role of interaction in information visualization," *IEEE Tran. on Visualization and Computer Graphics ,* vol. 13, no. 6, pp. 1224-1231, 2007.

[15] P. Lyman, H. R. Varian, P. Charles, N. Good, L. L. Jordan and J. Pal, "How much information? 2003," University of California at Berkeley, 2003.

[16] R. E. Bellman, Dynamic programming, Princeton University Press, 1957.

[17] D. A. Keim and H. P. Kriegel, "Visualization techniques for mining large databases: A comparision," *IEEE Trans. on Knowledge and Data Engineering,* vol. 8, no. 6, pp. 923-938, 1996.

[18] M. d'Ocagne, Coordonnées parallèles et axiales : Méthode de transformation géométrique et procédé nouveau de calcul graphique déduits de la considération des coordonnées parallèles, Paris: Gauthier-Villars, 1885.

[19] D. B. Carr, R. J. Littlefield, W. L. Nicholson and J. S. Littlefield, "Scatterplot matrix techniques for large N," *Journal of the Americal Statistical Association,* vol. 82, no. 398, pp. 424-436, 1987.

[20] M. Friendly and D. Denis, "The early origins and development of the scatterplot," *Journal of the History of the Behavioral Sciences,* vol. 41, no. 2, pp. 103-130, 2005.

[21] R. Rao and S. Card, "The table lens: Merging graphical and symbolic representation in an interactive focus+context visualization for tabular information," in *Proc. of SIGGHI Conference on Human Factors in Computing Systems*, p.318-322, 1994.

[22] H. Sagan, Space-filling curves, New York: Springer-Verlag, 1994.

[23] D. Hilbert, "Über die stetige Abbildung einer Linie auf ein Flächenstück," *Mathematische Annalen,* vol. 38, pp. 459-460, 1891.

[24] N. Boccara, Essentials of Mathematica: with applications to mathematics and physics, Springer, 2007.

[25] E. Kandogan, "Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions," in *In Proc. of IEEE Information Visualization Symp. (Hot Topics)*, pp. 9-12, 2001.

[26] N. Cao, D. Gotz, J. Sun and H. Qu, "DICON: interactive visual Aanalysis of multidimensional clusters," *IEEE Trans. on Visualization and Computer Graphics,* vol. 17, no. 12, pp. 2581-2590, 2011.

[27] B. Shneiderman, "Tree visualization with treemaps: A 2-d space-filling approach," *ACM Transaction on Graphics,* vol. 11, no. 1, pp. 92-99, 1992.

[28] "SmartMoney," [Online]. Available: http://www.marketwatch.com.

[29] M. Bruls, K. Huizing and J. J. Wijk van, "Squarified treemaps," in *In proc. of Joint Eurographics & IEEE TCVG Symp. on Visualization*, pp. 33-42, 2000.

[30] M. Balzer and O. Deussen, "Voronoi treemap," in *IEEE Symp. on Information Visualization*, Minneapolis, USA, pp 49-56, 2005.

[31] H. Siirtola and K. J. Raiha, "Interacting with parallel coordinates," *Interacting with Computers,* vol. 18, no. 6, pp. 1278-1309, 2006.

[32] H. Hauser, F. Ledermann and H. Doleisch, "Angular brushing of extended parallel coordinates," in *IEEE Symp. on Information Visualization*, Washington DC, USA, pp. 127-130, 2002.

[33] H. Zhou, X. Yuan, H. Qu, W. Cui and B. Chen, "Visual clustering in parallel coordinates," *Computer Graphics Forum,* vol. 27, no. 3, pp. 1047-1054, 2008.

[34] A. O. Artero, M. C. F. De Oliveira and H. Levkowitz, "Uncovering clusters in crowded parallel coordinates visualizations," in *In proc. of IEEE Symp. on Information Visualization*, pp. 81-88, 2004.

[35] Y. H. Fua, M. O. Ward and E. A. Rundersteiner, "Hierarchical parallel coordinates for exploration of large datasets," in *In Proc. of the conference on Visualization*, pp. 43-50, 1999.

[36] M. Ward, "XmdvTool: integrating multiple methods for visualizing multivariate data," in *In Proc. Visualization*, pp. 326-333, 1994.

[37] M. Novotny and H. Hauser, "Outlier-preserving focus+context visualization in parallel coordinates," *IEEE Tran. on Visualization and Computer Graphics,* vol. 12, no. 5, pp. 893-900, 2006.

[38] M. Ankerst, S. Berchtold and D. A. Keim, "Similarity clustering of dimensions for an enhanced visualization of multidimensional data," in *In Proc. of IEEE Symposium on Information Visualization*, pp. 52-60, 1998.

[39] W. Peng, M. Ward and . E. Rundersteiner, "Clutter reduction in multi-dimensional data visualization using dimension reordering," in *IEEE Symposium on Information Visualization*, pp. 89–96, 2004.

[40] J. Yang, W. Peng, M. O. Ward and E. A. Rundersteiner, "Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets," in *IEEE Symp. on Information Visualization*, Seattle, WA, USA, p.105-112, 2003.

[41] T. H. Huang, M. L. Huang and J. S. Jin, "Parallel rough set: dimensionality reduction and feature discovery of multi-dimensional data in visualization," in *International Conference on Neural Information Processing*, pp. 99-108, 2011.

[42] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Berkeley Symposium on Mathematical Statistics and Probability*, (1), pp. 281-297, 1967.

[43] J. A. Hartigan and M. A. Wong, "A K-means clustering algorithm," *Journal of Royal Statistics Society,* vol. 28, no. 1, pp. 100-108, 1979.

[44] M. B. Eisen, P. T. Spellman, P. O. Brown and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," in *Proc. Natl. Acad. Sci.*, 1998.

[45] J. Yang, M. O. Ward, E. A. Rundersteiner and S. Huang, "Visual hierarchical dimension reduction for exploration of high dimensional datasets," in *Joint Eurographics/IEEE TCVG Symposium on Visualization*, p.19-28, 2003.

[46] S. Johansson and 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.

[47] I. K. Fodor, "A survey of dimensionality reduction techniques," UCRL-ID-148494, LLNL Technical Report, 2002.

[48] H. F. Kaiser, "The application of electronic computers to factor analysis," *Educational and Psychological Measurement,* vol. 20, pp. 141-151, 1960.

[49] R. B. Cattell, "The scree test for the number of factors," *Multivariate Behavioral Research,* vol. 1, no. 2, pp. 245-276, 1966.

[50] J. Friedman and J. tukey, "A projection pursuit algorithm for exploratory data analysis," *IEEE Tran. Computers,* vol. 23, no. 9, pp. 881-890, 1974.

[51] P. J. Huber, "Projection pursuit," *The annual of Statistics,* vol. 13, no. 2, pp. 435-475, 1985.

[52] D. F. Swayne, N. Hubbell and A. Buja, "XGobi meets S: integrating software for data analysis," in *Comp. Sci. and Stat.: Proc. 23 rd Symp. Interface*, pp. 430-434, 1991.

[53] Z. Pawlak, Rough Set: Theoretical aspects of reasoning about data, Kluwer. Netherlands, 1991.

[54] D. Norman, "Twelve issues for cognitive science," *Cognitive Science,* no. 4, pp. 1-32, 1980.

[55] Z. Liu and J. T. Stasko, "Mental models, visual reasoning and interaction in information visualization: a top-down perspective," *IEEE Tran. on Visualization and Computer Graphics,* vol. 16, no. 6, pp. 999-1008, 2010.

[56] B. Shneiderman, "The eyes have it: a task by data type taxonomy for information visualizations," in *IEEE Symposium on Visual Languages*, pp. 336-343, 1996.

[57] E. H. Chi and J. T. Riedl, "An operator interaction framework for visualization systems," in *IEEE Symp. on Information Visualization*, pp. 63-70, 1998.

[58] M. C. Chuah and S. F. Roth, "On the semantics of interactive visualizations," in *IEEE Symp. on Information Visualization*, pp. 29-36, 1996.

[59] M. L. Huang and P. Eades, "A fully animated interactive system for clustering and navigating huge graphs," in *Graph Drawing. Vol. 1547 of LNCS*, Springer Berlin Heidelberg, 1998, pp. 374-383.

[60] M. Theus, "Interactive data visualization using Mondrian," *Journal of Statistical Software,* vol. 7, no. 11, pp. 1-9, 2002.

[61] J. Blaas, C. P. Botha and F. H. Post, "Extensions of parallel coordinates for interactive exploration of large multi-timepoint data sets," *IEEE Tran. on Visualization and Computer Graphics,* vol. 14, no. 6, pp. 1436-1443, 2008.

[62] C. A. Steed, P. J. Fitzpatrick, T. J. Jankun-Kelly, A. N. Yancey and J. E. Swan II, "An interactive parallel coordinates technique applied to a tropical cyclone climate analysis," *Computers & Geosciences,* vol. 35, no. 7, pp. 1529-1539, 2009.

[63] H. Siirtola, "Direct manipulation of parallel coordinates," in *IEEE International Conference on Information Visualization*, London, pp. 373-378, 2000.

[64] J. Alsakran, Y. Zhao and X. Zhao, "Tile-based parallel coordinates and its application in financial visualization," in *In Proceedings of Visualization and Data Analysis 2010*, San Jose, California, 2010.

[65] R. Shannon, T. Holland and A. Quigley, "Multivariate graph drawing using parallel coordinate visualisations," University College Dublin, 2008, Technical Report UCD-CSI-2008-06.

[66] P. Guo, H. Xiao, Z. Wang and X. Yuan, "Interactive local clustering operations for high dimensional data in parallel coordinates," in *IEEE Pacific Visualization Symposium*, pp. 97-104, 2010.

[67] Y. D. Liang and B. Barsky, "A new concept and method for line clipping," *ACM Transactions on Graphics,* vol. 3, no. 1, pp. 1-22, 1984.

[68] H. Lam, "A framework of interaction costs in information visualization," *IEEE Transaction on Visualization and Computer Graphics,* vol. 14, no. 6, pp. 1149-1156, 2008.

[69] P. Michaud, "Clustering technique," *Future Generation Computer Systems,* vol. 13, no. 2-3, pp. 135-147, 1997.

[70] W. H. E. Day and H. Edelsbrunner, "Efficient algorithms for agglomerative hierarchical clustering method," *Journal of Classification,* vol. 1, no. 1, pp. 7-24, 1984.

[71] T. H. Cormen , C. E. Leiserson , R. L. Rivest and C. Stein, "Introduction to algorithms," in *Introduction to algorithms*, MIT Press, 2009, pp. 540-549.

[72] F. Yamaguchi, Curves and surfaces in computer aided geometric design, Berlin, Germany: Springer-Verlag, 1988.

[73] G. G. Lorentz, Bernstein polynomials, University of Toronto Press, 1953.

[74] M. Graham and J. Kennedy, "Using curves to enhance parallel coordinate visualisations," in *In Proc. of International Information on Information Visualization*, pp. 10-16, 2003.

[75] T. W. Sederberg, "Computer aided geometric design," CAGD Course Notes, 2007.

[76] D. Freedman and P. Diaconis, "On the histogram as a density estimator:L 2 theory," *Probability Theory and Related Fields,* vol. 57, no. 4, pp. 453-476, 1981.

[77] H. A. Sturges, "The choice of a class interval," *Journal of the American Statistical Association,* vol. 21, no. 153, pp. 65-66, 1926.

[78] "OpenTK," [Online]. Available: http://www.opentk.com/.

[79] D. A. Keim, F. Mansmann, A. Stoffel and H. Ziegler, "Visual Analytics," in *Encyclopedia of Database Systems*, Springer, 2009, pp. 3341-3346.

[80] P. Thomas and T. Duff, "Compositing digital images," *Computer Graphics,* vol. 18, no. 3, pp. 253-259, 1984.

[81] C. Stolte, D. Tang and P. Hanrahan, "Multiscale visualization using data cubes," *IEEE Transactions on Visualization and Computer Graphics,* vol. 9, no. 2, pp. 176-187, 2003.

[82] A. Cockburn , A. Karlson and B. B. Bederson , "A review of overview+detail, zooming, and focus+context interfaces," *ACM Computing Surveys (CSUR),* vol. 41, no. 1, p. Article No. 2, 2008.

[83] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," *Annuals of Mathematical Statistics,* vol. 27, no. 3, pp. 832-837, 1956.

[84] E. Parzen, "On estimation of a probability density function and mode," *The Annals of Mathematical Statistics,* vol. 33, no. 3, pp. 1065-1076, 1962.

[85] W. Hardle and O. Linton, "Applied nonparametric methods," Cowles Foundation Discussion Papers 1069, Cowles Foundation for Research in Economics, Yale University, 1994.

[86] B. W. Silverman, Density estimation for statistics and data analysis, London: Chapman & Hall, 1986.

[87] M. P. Wand and M. C. Jones, Kernel smoothing, London: Chapman and Hall, 1995.

[88] S. J. Sheather and M. C. Jones, "A reliable data-based bandwidth selection method for kernel density estimation," *Journal of the Royal Statistical Society,* vol. 53, no. 3, p. 683–690, 1991.

[89] M. Rudemo, "Empirical choice of histograms and kernel density estimators," *Scandinavian Journal of Statistics,* vol. 9, no. 2, p. 65–78, 1982.

[90] M. Ankerst, D. A. Keim and H. P. Kriegel, "Circle Segments: a technique for visually exploring large multidimensional data sets," in *In IEEE Visualization, Hot Topic Session*, San Francisco, CA, 1996.

[91] K. Pearson, "Mathematical contributions to the theory of evolution. III. Regression, heredity and panmixia," *Phil. Trans. R. Soc. Lond. A,* vol. 187, pp. 253-318, 1896.

[92] W. S. Torgerson, "Multidimensional Scaling, I: theory and method," *Psychometrika,* vol. 17, pp. 401-419, 1952.

[93] F. Wickelmaier, "An Introduction to MDS," Sound Quality Research Unit, Aalborg University, Denmark, 2003.

[94] W. Ziarko, "Variable precision rough set model," *Journal of Computer & System Science,* vol. 46, no. 1, pp. 39-59, 1993.

[95] R. Rosenholtz, Y. Li and L. Namano, "Measuring visual clutter," *Journal of Vision,* vol. 7, no. 2, pp. 1-22, 2007.

[96] D. F. Swayne, A. Buja and D. T. Lang, "Exploratory visual analysis of graphs in GGobi," in *COMPSTAT 2004 - Proceedings in Computational Statistics*, pp. 477-488, 2004.

[97] "The Big List of D3.js Examples," [Online]. Available: http://christopheviau.com/d3list/. [Accessed 2014].

[98] "D3.js - Data-Driven Documents," [Online]. Available: http://d3js.org/. [Accessed 2014].

[99] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems,* vol. 47, no. 4, pp. 547-553, 2009.

[100] E. R. Berndt, The practice of econometrics, NY: Addison-Wesley, 1991.

[101] T. H. Huang, M. L. Huang and K. Zhang, "An interactive scatter plot metrics visualization for decision trend analysis," in *ICMLA*, pp. 258-264, 2012.

[102] Y. H. Chan, C. D. Correa and L. K. Ma, "Flow-based scatterplots for sensitivity analysis," in *In proc. IEEE Symp. on VAST*, pp. 43-50, 2010.

[103] S. Chatterjee and A. S. Hadi, Sensitivity analysis in linear regression, Wiley, 1988.

[104] S. G. AKL and G. T. Toussaint, "A fast convex hull algorithm," *Information Processing Letters,* vol. 7, no. 5, pp. 219-222, 1978.

[105] 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,* vol. 1, no. 2, pp. 20-34, 2002.

[106] C. G. Healey , "Choosing effective colours for data visualization," in *IEEE Conference on Visualization (VIS'96)*, pp. 263-271, 1996.

[107] P. Bourke, "Colour Spaces," [Online]. Available: http://paulbourke.net/texture_colour/colourspace/.

[108] M. Harrower and C. A. Brewer, "ColorBrewer.org - an online tool for selecting colour schemes for maps," *The Cartographic Journal,* vol. 40, no. 1, pp. 27-37, 2003.