

Visualizing High-density Clusters in Multidimensional Data

Tran Van Long

A thesis submitted in partial fulfilment
of the requirements for the degree of

Doctor of Philosophy
in
Computer Science

Approved, Thesis Committee:

Prof. Dr. Lars Linsen (supervisor)

Prof. Dr. Adalbert F. X. Wilhelm

Prof. Dr. Daniel Keim

Date of Defense: December 11, 2009

School of Engineering and Science

To my parents

Abstract

The analysis of multidimensional multivariate data has been studied in various research areas for many years. The goal of the analysis is to gain insight into the specific properties of the data by scrutinizing the distribution of the records at large and finding clusters of records that exhibit correlations among the dimensions or variables. As large data sets become ubiquitous but the screen space for displaying is limited, the size of the data sets exceeds the number of pixels on the screen. Hence, we cannot display all data values simultaneously. Another problem occurs when the number of dimensions exceeds three dimensions. Displaying such data sets in two or three dimensions, which is the usual limitation of the displaying tools, becomes a challenge.

To overcome these two limitations, this dissertation proposes a framework that can help analyzers to understand the distribution of multidimensional multivariate data sets. It supports discovering clusters, patterns, outliers, and relationships.

The main approach consists of two major steps: clustering and visualizing. In the clustering step, we examine the structure of the multidimensional multivariate data sets by their density distributions. Based on the density distribution of a data set, we propose two clustering algorithms to construct hierarchical density clusters. In the visualizing step, we propose two methods to visually analyze the hierarchical density clusters. An optimized star coordinates approach is used to project high-dimensional data into the (two- or three-dimensional) visual space, in which the leaf clusters of hierarchical density clusters (well-separated in the original data space) are projected into visual space with minimizing the overlapping. Each cluster is displayed by an enclosing contour or surface. The second method, we developed to visualize the hierarchical density cluster tree, combines several information visualization techniques in linked and embedded displays: radial layout for hierarchical structures, linked parallel coordinates, and embedded circular parallel coordinates.

By combining cluster analysis with star coordinates or parallel coordinates, we extend these visualization techniques to cluster visualizations. We display clusters instead of data points. The advantage of this combination is scalability with both the size and dimensions of data set.

Acknowledgements

I want to express my thank to all people who supported me during three years while I have been working on my thesis studying.

First of all, I want to give my most grateful to Professor Dr. Lars Linsen who creates favorable conditions for studying at Jacobs University, Bremen, Germany. I would like to thank him, my advisor, for his fruitful scientific guidance, useful discussions, comments, and suggestions.

I am grateful to my colleagues Sherin Al-Shbat, Steffen Hauth, Tetyana Ivanovska, and Paul Rosenthal for creating a pleasant working atmosphere. Special thanks go to Paul Rosenthal for his cooperation in visualization contest 2008 and useful discussions.

I am also thankful to my friends Tran Manh Ha, Tran Que Tien, Tran Hong Minh at Jacobs University Bremen, and Dang Duc Hanh, Huynh Anh Huy, Nguyen Manh Thang at University Bremen, who make my life more fun and less boring.

Finally, I would like to express my gratefulness to my wife for her love and patience.

Tran Van Long
Bremen, October 2009

Contents

1	Introduction	1
1.1	Data visualization	1
1.2	Multidimensional data visualization	3
1.3	Goals	7
1.4	Overview	8
1.4.1	Construction of hierarchical density clusters	8
1.4.2	Visualization of hierarchical density clusters	8
1.5	Contributions	9
1.6	Structure	10
2	Related work	11
2.1	Information visualization	11
2.1.1	Concepts and terminology	11
2.1.2	Visualization techniques	12
2.1.3	Star coordinates	16
2.1.4	Parallel coordinates	19
2.2	Cluster analysis	23
2.2.1	Hierarchical clustering	24
2.2.2	Hierarchical density clusters	28
2.3	Visualization of hierarchical clustering results	30
2.3.1	Visualizing hierarchical structures	30
2.3.2	Visualizing hierarchical density clusters	31
3	Hierarchical density clusters	35
3.1	Nonparametric density estimation	35
3.1.1	Nonparametric estimation criteria	35
3.1.2	Multivariate histogram density estimation	36
3.1.3	Multivariate kernel density estimation	39
3.2	Hierarchical density clusters	42
3.2.1	Hierarchical density clusters using histograms	43
3.2.2	Hierarchical density clusters using kernels	47
3.2.3	Discussions and comparisons	51
4	Nested level set visualization of hierarchical density clusters	55
4.1	Optimized star coordinates	55
4.2	Cluster enclosure	59

4.2.1	Euclidean minimum spanning tree	60
4.2.2	Density field functions	61
4.2.3	Enclosing point clouds	63
4.3	Results and discussions	65
5	Interactive visual exploration of hierarchical density clusters	75
5.1	Radial layout of density cluster hierarchy	75
5.2	Linked views with parallel coordinates	78
5.3	Integrating circular parallel coordinates	83
5.4	Case study	86
6	Conclusion and future work	95
	References	99

Chapter 1

Introduction

The large amount of information available today bears an enormous potential. Thus, it gets more and more important to find ways to determine and present sets of data, which are relevant for a specific task. Card et al. [CMS99] give the following definition:

Information visualization is the use of computer-supported, interactive, visual representations of abstract data to amplify cognition.

Information visualization deals with the general problem of representing abstract data. The goal of presentations is to help users understanding the data. The data is transformed into an image, it is mapped to screen space. The image can be changed by users as they proceed working with it. This interaction is important as it allows for constant redefinition of goals when new insight into the data has been gained.

1.1 Data visualization

Data visualization differs little from information visualization. The data in data visualization does not consist of abstract information, it is usually a real world situation. Data visualization is the process of using graphical presentation to represent complex data in a way that provides the viewer with a qualitative understanding of its information contents, turning complicated sets of data into visual insights.

Data visualization solutions bring clarity to numerical data through visual representation helping to reveal insights and trends that might otherwise be unnoticed. The challenge is to find a suitable visualization technique to give more insight into the data. The major goals of data visualization are presentation, confirmative analysis, and explorative analysis [Kei97].

Presentation If everything is known about the data, the first important data visualization goal is to present the data. In this case, data visualization serves for communication of the results. For example, Figure 1.1 shows the terrible fate of Napoleon's army in Russia [Tuf83]. Beginning at left, the thick tan flow-line shows the size of the Grand Army. The width of this band indicates the size of the army at each place on the map. The path of Napoleon's retreat from Moscow is depicted by the darker, lower band. This figure display several variables: the size of the army, its location on a two-dimensional surface,

direction of the army's movement, and temperature on various dates during the retreat from Moscow.

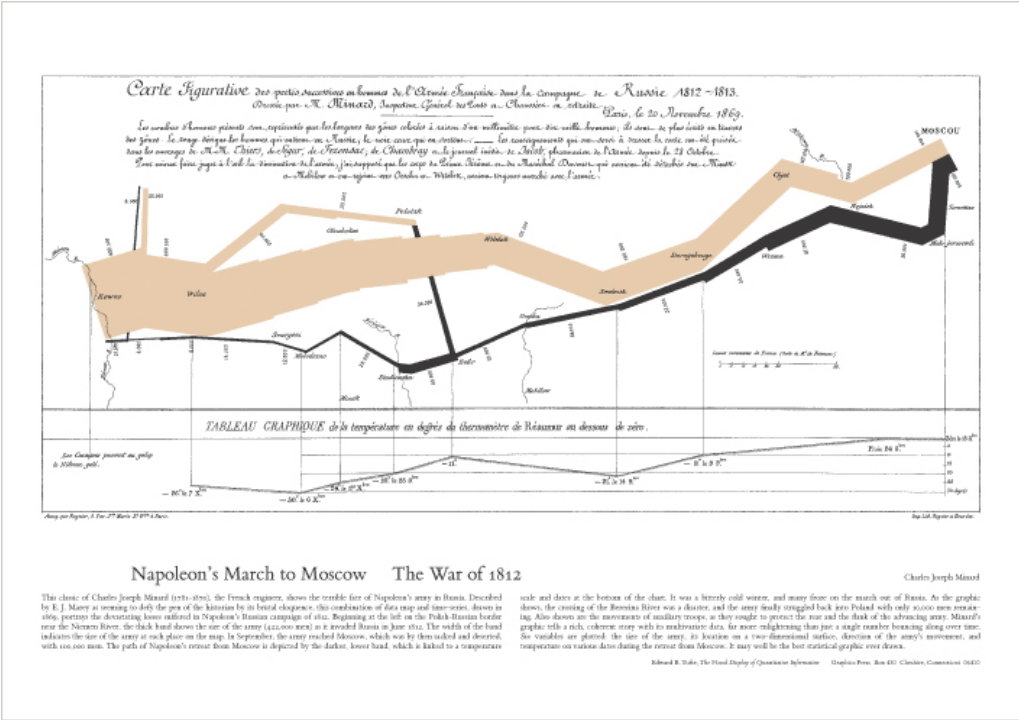


Figure 1.1: Minard's map of Napoleon's Russian campaign.

Exploration analysis Data visualization can be used to create hypotheses. Typically, we know a priori very little about the data. In this case, data visualization helps searching for structures, trends, and outliers. For example, Figure 1.2 is plotted by Snow [Tuf83] that shows the locations of death from cholera in central London on September 1854. Deaths were marked by dots. Examining the scatter over the surface of the map, Snow observed that cholera occurred almost entirely among those who lived near the Broad Street water pump. Exploratory data analysis [NIS03] is an approach/philosophy for data analysis that employs a variety of techniques (mostly graphical) to

- maximize insight into a data set,
- uncover underlying structure,
- extract important variables,
- detect outliers and anomalies,
- test underlying assumptions,
- develop parsimonious models, and
- determine optimal factor settings.

Confirmative analysis If there are hypotheses about the data, data visualization can be used for examination. The goal is to verify or to disprove these hypotheses. This also includes quality control of simulations and measurements.



Figure 1.2: A map of cholera deaths in London, 1840.

1.2 Multidimensional data visualization

Multidimensional multivariate visualization is an important subfield of data visualization that focuses on multidimensional multivariate data sets. Multidimensional multivariate data can be defined as a set of observations X , where the i th element x_i consists of a vector with m variables, $x_i = (x_{i1}, \dots, x_{im})$. Each variable may be independent or dependent on one or more other variables. Independent variables are referred to multidimensional variables and dependent variables are referred to multivariate [WB97].

Visual exploration of multidimensional multivariate data is of great interest in both statistics and information visualization. It helps the user to find trends, patterns, outliers, and relationships among variables. When visualizing multidimensional multivariate data, each variable may map to some graphical entity or attribute. According to the different ways of dimensionality manipulation, we can broadly categorize the display techniques as:

- Axis reconfiguration techniques map directly each multidimensional data point to a glyph, such as parallel coordinates [Ins85, Weg90] and glyphs [And72,

Che73].

- Dimensional embedding techniques present subspace of multidimensional data space in a hierarchical fashion, such as dimensional stacking [LWW90] and worlds within worlds [FB90].
- Dimensional subsetting techniques map the attributes to Cartesian coordinates, such as scatterplot matrix [Cle93], hyperslide [vWvL93] and hyper-box [AC91].
- Dimensional reduction techniques map multidimensional data into a space of lower dimensions with preserving relationships of the multidimensional data, such as multidimensional scaling [BG05], principal component analysis [Jol86], and self-organizing maps [Koh95].

The most common visualization techniques to present multidimensional data are the scatterplot matrix and parallel coordinates. We present these visualization techniques with a well known data set, called *iris* data set. The iris data set contains 150 data points with four attributes: sepal length, sepal width, petal length, and petal width. The iris data set consists of 50 observations from each of three species of iris flowers: iris setosa, iris virginica, and iris versicolor. Figure 1.3 shows this data set with a scatterplot matrix and Figure 1.4 shows this data set with parallel coordinates. In both scatterplot matrix and parallel coordinates, we can identify group setosa well separated from the two other groups. The two groups virginica and versicolor are highly mixed with two attributes: sepal length and sepal width and less mixed with the two other attributes: petal length and petal width. The group virginica has larger values for the attributes petal length and petal width than the group versicolor.

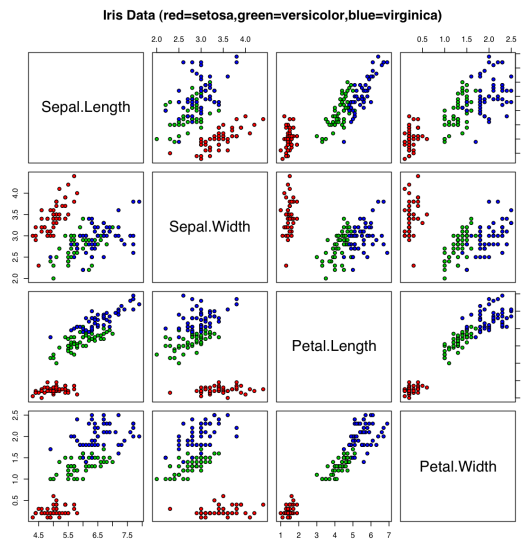


Figure 1.3: Scatterplot matrix presents the iris data set.

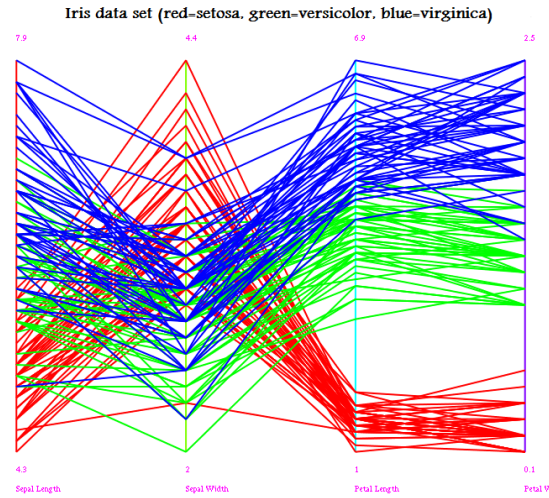


Figure 1.4: Parallel coordinates represent the iris data set.

We also consider a synthetic data set with a few hundreds of data points and high dimensionality. This data set consists of 480 data points with ten dimensions and contains 14 clusters. Figure 1.5 shows this data set using a scatterplot matrix. There are a large number of scatterplots. In each scatterplot, we can only identify five clusters, some clusters are overlapping. We cannot display 14 clusters with a scatterplot matrix. Additionally we display this data set with parallel coordinates, Figure 1.6 shows 14 clusters, but only the middle cluster is obvious. Because of overplotting in parallel coordinates, it is difficult to see the other clusters.

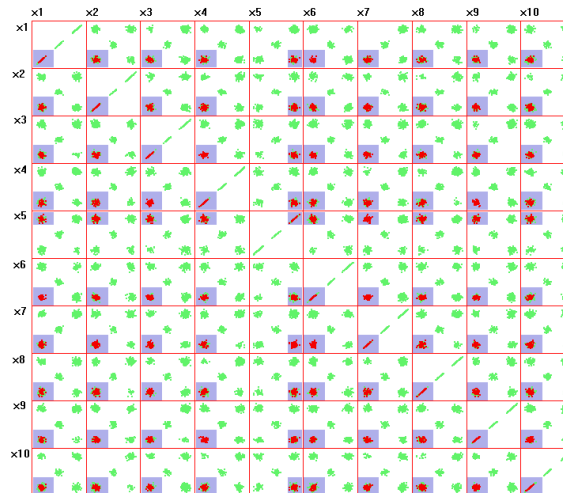


Figure 1.5: A synthetic data set is displayed in the scatterplot matrix.

As large data sets become more and more common, it has become clear that most existing multidimensional multivariate techniques lose their effectiveness when more than a few hundred or thousand data points are displayed. The reason is that the available screen space is limited. As a result, the clutter problem becomes a serious issue in the visualization of large multidimensional multivariate data sets.

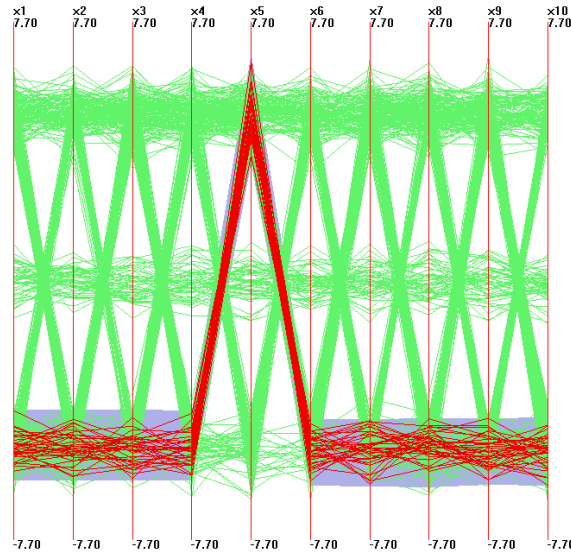


Figure 1.6: A synthetic data set is displayed in parallel coordinates.

For example, Figure 1.7 shows the parallel coordinates visualization of one of the most popular data sets: the *out5d* data set containing 16384 data points with five dimensions. The data set contains five clusters, but we cannot see any clusters in this figure.

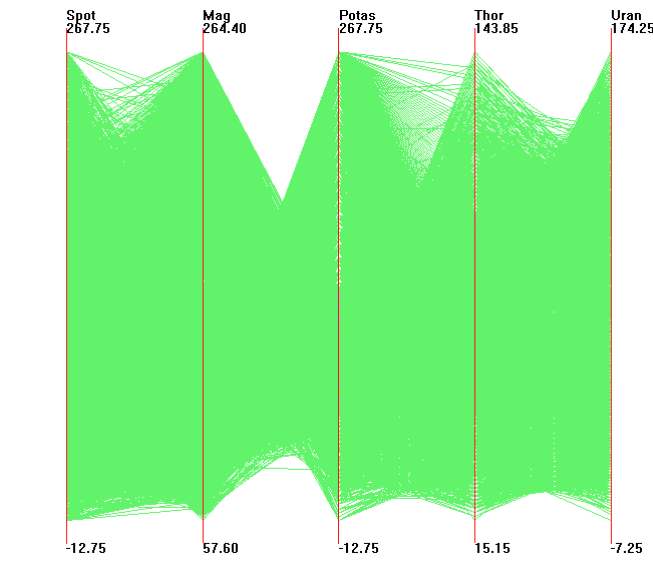


Figure 1.7: Out5d data set is visualized in parallel coordinates.

Multidimensional data sets are often dealing with huge size and high dimensionality. It is difficult to visualize such data in a single visual space. One major challenge of multidimensional data visualization is to display intuitive structures of the data set with all attributes simultaneously, but avoid cluttering, i.e., overlapping.

1.3 Goals

Clustering enables researchers to see overall distribution patterns, identify interesting unusual patterns, and spot potential outliers. Cognition of the clustering results can be amplified by dynamic queries and interactive visual representation methods. Understanding of the clustering results is transformed to another important data mining task - exploratory data analysis. Interactive information visualization techniques enable users to effectively explore clustering results and help them to find the informative clusters that lead to insights.

Nonparametric density estimations have been successfully applied in exploratory data analysis for one- and two-dimensional data. For example, it is possible to detect modes by inspection of one- and two-dimensional density estimates. For higher-dimensional data the difficulties with visualizing density estimates have often hindered the application of nonparametric density estimation. We construct methods for visualization of multivariate functions, which amplify usefulness of multivariate density estimates in exploration and mining of multidimensional multivariate data.

When dealing with large data sets with many observations, clustering has proven to be extremely useful. Clustering is a partition method of a data set into subsets of similar observations. Each subset is called a cluster, which consists of observations that are similar within themselves and dissimilar to observations of other clusters. Cluster analysis tasks for multidimensional data have the goal of finding areas where the observations group together to form a cluster.

The structure of a data set is reflected by its density function. In this thesis, we use a definition of the multidimensional multivariate data cluster based on a multivariate density function. Hartigan [Har75, Har81] defined the concept of a *high density cluster*: a high density cluster at level λ in a population with density function $p(x)$ is defined as a maximal connected set of points x with $p(x) \geq \lambda$, where λ is constant and positive. For various levels λ , we obtain the collection of high density clusters \mathcal{T} , that has a hierarchical structure: for any two high density clusters A and B in \mathcal{T} , we have $A \subset B$ or $B \subset A$ or $A \cap B = \emptyset$. This hierarchical structure is summarized by the high density cluster tree of the probability density function $p(x)$. Leaf nodes of the hierarchical density cluster are corresponding to clusters that are homogeneous, i.e., contain only one mode, and internal nodes of the hierarchical density cluster are corresponding to clusters that are heterogeneous, i.e., contain more than one mode.

The basic goals of this thesis are to:

- present the concept of high density clusters and hierarchical high density clusters,
- develop hierarchical clustering algorithms to construct hierarchical density clusters for large high dimensional data sets,
- visualize hierarchical density clusters by nested density level sets, and
- build a system that integrates a hierarchical structure visualization with multidimensional multivariate visualization techniques.

1.4 Overview

A hierarchical density cluster of a multidimensional multivariate data set reflects the mode structures of the density function, as the different subtrees of the hierarchical density cluster correspond to distinct regions that contain distinct modes.

1.4.1 Construction of hierarchical density clusters

A hierarchical density cluster of a given data set is created based on density estimation. The density estimator is computed based on a grid partitioning in multidimensional data space (histogram methods, i.e., the multidimensional data space is quantized into a finite number of cells that form a grid structure) or adaptive intersecting partitioning of the multidimensional data space (kernel methods).

A level set of a density function $p(x)$ at a given density value λ is a set $\{x : p(x) \geq \lambda\}$. The set $\{x : p(x) \geq \lambda\}$ can partition into several connected components. The connected components are clusters at the density value λ . The complex data set is reduced recursively by increasing level of density values and by clustering the data set in multidimensional space to form a hierarchical structure. The hierarchical structure forms a tree, where the root node of the tree is corresponding to the density value $\lambda = 0$. The leaf nodes represent homogeneous clusters, i.e., clusters contain only one mode of the density distribution and the internal nodes represent heterogeneous clusters, i.e., clusters contain more than one mode of the density distribution. At each level set, clusters form maximally connected components of that level of density and clusters are separated by regions of lower level of density.

1.4.2 Visualization of hierarchical density clusters

One of the commonly used methods to cope with high dimensionality is to use low-dimensional projections. Since human eyes and minds are effective in understanding one-dimensional data with histogram representations, two-dimensional data with 2D scatterplot representations, and three-dimensional data with 3D scatterplot representations, these representations are often used as a starting point. Users can begin by understanding the meaning of each dimension and by examining the range and distribution of the values in the histogram. Users can explore two-dimensional or three-dimensional relationships by studying 2D or 3D scatterplots. Collections of 2D or 3D projections have been widely used as representations of the original multidimensional data. This is imperfect since some features may be hidden, but at least users can understand what they are seeing and obtain some insights.

We propose a method to project multidimensional data sets to a 2D or 3D visual space. The projection method uses an optimized star coordinates layout. The optimization procedure minimizes the overlap of projected homogeneous clusters in the hierarchical density clusters. This projection is a linear and contracting mapping. The star coordinates visualization allows for an interactive analysis of the distribution of clusters and comprehension of the relations between clusters and the original dimensions. Clusters in visual space can be displayed by extracting contours or surfaces enclosing the set of points inside these clusters. The hierarchical

density clusters are visualized by nested sequences of density level sets leading to a quantitative understanding of information content, patterns, and relationships. This approach is presented in [LLRR08, LL09b].

Studies on multidimensional data analysis led us to design and implement an interactive visualization tool called MultiClusterTree [LLR09, LL09a]. MultiClusterTree supports interactive exploration of hierarchical density clusters to enable users to build a good description understanding of the data sets. MultiClusterTree explores multidimensional data sets by visualizing high density clusters using a radial layout, providing linked views of the radial cluster tree with parallel coordinates and an integrated view of the radial cluster tree and circular parallel coordinates.

1.5 Contributions

This thesis addresses the problem of visual analysis of multidimensional multivariate data. Our approach is based on the analysis of the data's density distribution. We describe an interactive exploration system for multidimensional multivariate data analysis ranging from density computation over an automatic hierarchical density cluster computation to an optimized projection method into the visual space based on star coordinates, where clusters of the hierarchical density clusters are rendered using nested contours or surfaces. We also describe another approach to visualize the hierarchical density clusters with the concept of a radial layout of the hierarchical structure. Based on the radial layout of the hierarchical density clusters, the clusters can be explored interactively using parallel coordinates when being selected in the hierarchical density clusters. Furthermore, we integrate circular parallel coordinates into the radial layout of hierarchical density clusters, which allows for an understanding of both the overall cluster distribution and characteristics of these clusters.

The individual contributions of this thesis include:

1. Computing hierarchical density clusters in a multidimensional data space, which ensures that all clusters can be identified and separated.
2. Coupling an automatic multidimensional clustering based on
 - (a) efficient grid computation, or
 - (b) efficient intersecting partitioning

of the multidimensional space with the concept of the hierarchical density clusters, which leads to an automatic computation of multidimensional hierarchical density clusters without manual adjusting of level set thresholds.

3. Projection of hierarchical density clusters into visual space with an optimized star coordinates layout such that
 - (a) the overlap of clusters well separated in multidimensional data space is minimized, and

- (b) the shape, compactness, and distribution of clusters are maintained as much as possible.
- 4. Visualizing hierarchical density clusters as nested bounding contours in a star coordinates layout, which
 - (a) shows the distribution of all clusters, and
 - (b) allows to correlate the clusters to the original dimensions.
- 5. Visualizing hierarchical density clusters based on a 2D radial layout with an automatic color coding.
- 6. Linking the hierarchical density clusters visualization to other views including parallel coordinates where interaction mechanisms operate on cluster nodes and for correspondence. The simultaneous display of selected clusters in parallel coordinates allows for a comparison of clusters with respect to the given dimensions.
- 7. Integrating circular parallel coordinates into the hierarchical density cluster visualization, also supporting a focus + context technique. This integrated view allows for an understanding of the entire data set without overplotting, such that both the individual clusters with respect to the given dimensions and the overall cluster distribution are easily comprehensible.

1.6 Structure

This thesis is organized as follows. Chapter 2 covers related work in multidimensional multivariate visualization techniques, hierarchical density clustering, and visualizing hierarchical density clusters. Chapter 3 introduces two algorithms to create hierarchical density clusters. First, hierarchical density clusters are created in a top-down approach based on histogram density estimation. Second, hierarchical density clusters are created with a bottom-up approach based on kernel density estimation. Chapter 4 presents a method visualizing hierarchical density cluster based on optimized star coordinates and clusters wrapping by contours or surfaces. Chapter 5 proposes a radial layout method for the hierarchical density clusters, linked views with parallel coordinates, and integration with circular parallel coordinates. This dissertation concludes with possible future works and contributions in Chapter 6.

Chapter 2

Related work

In this chapter, we provide a broad overview of work related to with this dissertation. We discuss information visualization techniques, cluster analysis, and visualizing hierarchical clustering. In particular, the concepts of star coordinates and parallel coordinates are discussed in the subsection on information visualization. The concept of hierarchical clustering is discussed in the subsection on cluster analysis and data visualization techniques for visualizing hierarchical structures are discussed in the subsection on visualizing hierarchical clustering.

2.1 Information visualization

As technology progresses, the amount of data increases rapidly. Data become ubiquitous on modern lives and works. Several sources of data include biotechnology about human genomes, financial about stock markets, information consumers, image analysis, and engineering (multidimensional data sets are generated by measurements and/or simulations). This is going to cover the whole research in modern science and is a true challenge for data analysis. Most of the data contains valuable and useful information. How can one extract the valuable information hidden in the data?

Human beings look for structures, trends, anomalies, and relationship in the data. Visualization supports this by representing the data in various forms of different kinds of interactions. A visualization can provide a qualitative overview of large complex data sets, can summarize data sets, and can assist in identifying regions of interest and appropriate parameters for more focused quantitative analysis [FGW02]. The idea is to combine cognition and perception with the computational power of computers.

2.1.1 Concepts and terminology

Multidimensional data is encountered in various kinds of analysis and data comes in many formats; for convenience, the data needs to be transformed into abstracted relations. We consider it as being given in a form of a usual data matrix, i.e., a

rectangular array of real values with n rows and m columns

$$X = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{pmatrix},$$

where n is the size of the data set, m is the number of attributes, and x_{ij} is a real value.

Rows represent different observations or records and columns represent different dimensions or attributes. For example, in gene expression data the i th row represents a gene and the j th column represents an experiment condition, and entry element x_{ij} is a real value representing the gene expression level of the i th gene under the j th condition. We consider n records as n independent observations from a random vector in m -dimensional space $X = (X_1, \dots, X_m)$.

An item of the data is composed of *variables*. When such a data item is defined by more than one variable it is called a *multivariable* data item. Variables are frequently classified into two categories: *dependent* and *independent*. Statisticians use the term “response” for dependent variables and “predictor” for independent variables.

In mathematics, a function is defined as a relation between two terms called variables. If every value of x is associated with exactly one value of y , then y is said to be a function of x . The variable x is called the *independent variable*, and the variable y is called the *dependent variable* because its value depends on the value of x .

In many data mining data sets, it is unknown whether a particular variable or attribute or feature is dependent or independent of any other attributes. The dependent variable is called *variate*, and data containing more than one variate is called *multivariate*. The independent variable is called *dimension*, and data containing more than one dimension is called *multidimensional*. The more appropriate term is “multidimensional multivariate” in the field of information visualizations [WB97].

2.1.2 Visualization techniques

A classification of visualization techniques is not straightforward. Some techniques are combining several ideas and others are very specific to a certain application. In this subsection, we describe three taxonomies of visualization techniques. We look at an overview on taxonomies by Wong and Bergeron [WB97], Card et al. [CMS99], and we discuss more about visualization techniques through the taxonomy by Keim [Kei02].

Taxonomy of visualization techniques by Wong and Bergeron

Wong and Bergeron [WB97] classified multidimensional multivariate visualization techniques based on bivariate displays, multivariate displays, and animations.

Techniques based on bivariate displays include the fundamental bivariate displays and simultaneous views of bivariate displays. One of the most popular

multidimensional multivariate visualization techniques is the scatterplot matrix which presents all combination pairs of all dimensions and organizing them by a matrix [Cle93]. In a scatterplot matrix, every variate is treated identically. The idea of pairwise adjacencies of variables is also a basis for the hyperbox [AC91], the hierarchical axis [MGTS90], and the hyperslide [vWvL93].

Multivariate displays are the basis for many recently developed multidimensional multivariate visualization techniques, most of which use colorful graphics created by high-speed graphics computations. These techniques can broadly be categorized into five sub-groups:

- *Brushing* allows direct manipulation of a multidimensional multivariate visualization display. This technique is described for scatterplot matrices [Cle93].
- *Panel matrix* involves pairwise two-dimensional plots of adjacent variates. These techniques include hyperslide [vWvL93] and hyperbox [AC91].
- *Iconography* uses variates to determine values of parameters of small graphical objects. The mappings of data values to graphical parameters are usually chosen to generate texture patterns that hopefully bring insight into the data. Some iconographic techniques are Chernoff face [Che73], stick figure icon [PG88], autoglyph [Bed90], and color icon [Lev91].
- *Hierarchical displays* map a subset of variates into different hierarchical levels of the display. Hierarchical axis [MGTS90], dimensional stacking [LWW90], and world within world [LWW90] visualization techniques belong to this group.
- *Non-Cartesian displays* map data into non-Cartesian axes. They include parallel coordinates [Ins85, ID90, Weg90] and visdb [KK94].

Animation is a powerful method for visualizing multidimensional multivariate scientific data. Various movie animation techniques on multidimensional multivariate data, and a scalar visualization animation model is presented. The most popular animation technique is the grand tour technique, in which multidimensional multivariate data is projected into two dimensional planes.

Taxonomy of visualization techniques by Card et al.

Card et al. [CMS99] introduced four ways to encode abstract data, a common occurrence in information visualization:

1D, 2D, 3D refers to orthogonal visualization that encodes information by positioning marks on orthogonal axes.

Multiple dimensions refer to the harder problem of multidimensional visualization where the data has so many variables that an orthogonal visual structure is not sufficient. Typical tasks that must be supported by such environments involve getting knowledge from the data, like finding patterns, relationships,

clusters, gaps, and outliers, or finding specific items using interaction, such as zooming, filtering, and selection.

Trees refer to using connection and enclosure to encode relationships among cases.

Networks refer to using connections to encode relationships among cases.

Taxonomy of visualization techniques by Keim

Keim [Kei02] classifies information visualization techniques by their basic visualization principle: geometric projection, iconographic, pixel-oriented, hierarchies, graph-based and hybrid.

Geometric projection techniques support users in the task of finding information projections of multidimensional multivariate data. In this way, a high number of dimensions can be visualized. Typical examples here are star coordinates and parallel coordinates, which are discussed in detail later in this chapter, and techniques included in the following Table 2.1:

Category	Visualization technique	References
Geometric projection	Scatterplot matrices	[Cle93]
	Andrews' plots	[And72]
	Projection pursuit	[FT74, Hub85]
	Parallel coordinates	[Ins85, ID90, Weg90]
	Prosection views	[FB94]
	Landscapes	[Wri95]
	Hyperslice	[vWvL93]
	Radviz	[HGM ⁺ 97]
	Star coordinates	[Kan00, Kan01]

Table 2.1: Geometric projection techniques of visualization techniques.

Iconographic display techniques map each multidimensional data item to an icon (or glyph) whose visual features vary depending on the data values. The number of displayable dimensions is not limited with this approach.

Category	Visualization technique	References
Iconographic	Stick figures	[Pic70, PG88]
	Chenoff faces	[Che73, Tuf83]
	Shape coding	[Bed90]
	Color icons	[Lev91]

Table 2.2: Iconographic techniques of visualization techniques.

However, they are not used very often for high-dimensional data sets, since a quick information exploration is problematic. The iconographical techniques are given in the Table 2.2.

Pixel-oriented In pixel-based techniques, a pixel is used to represent data values. Pixels are grouped according to the dimension, the item it belongs to, and are arranged on the screen appropriate to different purposes. In general, one pixel is used per data value, so the number of displayable values is rather high. The techniques are further categorized as “query independent” or “query dependent”. In the query independent techniques, the arrangement of the pixels in the subwindows is fixed, independently of the data values themselves. In the query dependent techniques, a query item is provided and distances from the data values to the given query value are computed using some metrics. The mapping of colors to pixels is based on the computed distances for each attribute and pixels in each subwindow are arranged according to their overall distances to the query data item. The Table 2.3 shows the pixel-oriented techniques.

Category	Visualization technique	References
Pixel-oriented	Circle segment	[AKK96]
	Spiral and axes techniques	[KK94]
	Recursive pattern	[KKA95]

Table 2.3: Pixel-oriented techniques of visualization techniques.

Hierarchical techniques subdivide the m -dimensional data space and represent subspaces in a hierarchical fashion. The hierarchical techniques are shown in the Table 2.4.

Category	Visualization technique	References
Hierarchies	Dimensional stacking	[LWW90]
	Worlds within worlds (n-vision)	[FB90]
	Conetrees	[RMC91]
	Treemap	[Shn92, Joh93]
	Infocube	[RG93]

Table 2.4: Hierarchical techniques of visualization techniques.

Graph-based techniques visualize large graphs using specific layout algorithms, query languages, and abstraction techniques to convey their meaning clearly and quickly. The graph-based techniques are given in the Table 2.5.

Category	Visualization technique	References
Graph-based	Hiernet	[EW93]
	Narcissus	[HDWB95]

Table 2.5: Graph-based techniques of visualization techniques.

2.1.3 Star coordinates

The method of star coordinates was introduced by Kadogan [Kan00, Kan01]. In star coordinates, each dimension is represented as a vector radiating from the center of a unit circle in a two-dimensional plane. Initially, all axes have the same length and are uniformly placed on the circle. Data points are scaled to the length of the axes, with the minimum being mapped to the origin and the maximum to the other end of the axes on the unit circle.

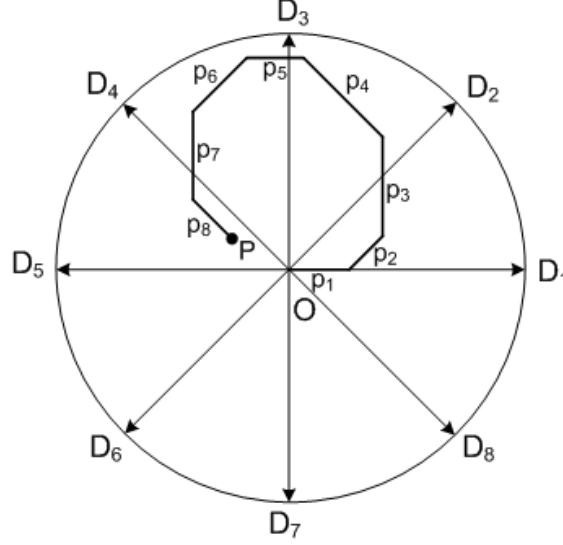


Figure 2.1: Calculation of data point location for an eight-dimensional data set.

In mathematics, the Cartesian coordinate system is used to determine each point uniquely in a plane through two numbers, usually called the x -coordinate and the y -coordinate. A point $P = (x, y)$ in the plane can be represented by a vector

$$P = O + x\mathbf{i} + y\mathbf{j},$$

where $\mathbf{i} = (1, 0)$, $\mathbf{j} = (0, 1)$ are the two basis vectors of the Cartesian coordinates and $O = (0, 0)$ is the origin.

A multidimensional point is represented in a plane similar to the Cartesian coordinates. The 2D star coordinates system is used for representing a point in m dimensions including m vectors in a plane

$$V = \{v_1, \dots, v_m\}.$$

Here $v_i = (v_{ix}, v_{iy}) = \left(\cos \frac{2\pi i}{m}, \sin \frac{2\pi i}{m} \right)$ is representing the i th dimension, $i = 1, \dots, m$, and the origin is $O = (O_x, O_y)$. A mapping of a point (p_1, \dots, p_m) to a point $P = (P_x, P_y)$ in two-dimensional Cartesian coordinates is determined by the sum of basic vectors $v_i = (v_{ix}, v_{iy})$ on each axis multiplied by the value of the point. More precisely, the formula is given by:

$$P = O + \sum_{i=1}^m p_i v_i,$$

or

$$\begin{cases} P_x = O_x + \sum_{i=1}^m p_i v_{ix}, \\ P_y = O_y + \sum_{i=1}^m p_i v_{iy}. \end{cases}$$

In Figure 2.1, the star coordinates system has eight axes D_1, \dots, D_8 represent the eight dimensions. These axes represent for basic vectors of the Cartesian coordinates that evenly placed on a unit disk. The point P in the two-dimensional space is on representation of the point in eight dimensions (p_1, \dots, p_8) . We can also explain geometrically how to find the point $P = O + \sum_{i=1}^m p_i v_i$: we start at the origin O of a circle, moving along the axis D_1 with length p_1 , continue moving parallel to the axis D_2 with length p_2 , and so on. The end point of this process is the point P .

All coordinates systems are given by an origin and some vectors. Typically, the vectors are linearly independent, e.g., Cartesian coordinates, and a point is uniquely represented. In the star coordinates system, the vectors are linearly dependent, and the representation of a point is not unique.

In general, the mapping from multidimensional space into a low-dimensional space is not unique. Only with an aid of interactive dynamic transformations such as rotations and translations one can make sense of the data representation. Star coordinates basically attempts to extend this idea to higher dimensions. Clusters, trends, and outliers in a data set are preserved in the projected multidimensional data visualization and interactions help to confirm this. Traditional star coordinates originally included rotation and scaling [Kan00] and Kandogan later extends it to include range selection, marking, histograms, footprints, and sticks [Kan01].

Artero and Oliveira introduced *Viz3D* that projects multidimensional data into a 3D display space [AdO04]. Similar to star coordinates, the basic system of *Viz3D* is obtained from the basic system of star coordinates by adding 1 to the third coordinates, that means the basic system of *Viz3D* is given by:

$$v_i = \left(\cos \frac{2\pi i}{m}, \sin \frac{2\pi i}{m}, 1 \right), i = 1, \dots, m$$

and the mapping from multidimensional data space into a 3D visual space is formulated as:

$$\begin{cases} P_x = O_x + \frac{1}{m} \sum_{i=1}^m p_i \cos \frac{2\pi i}{m} \\ P_y = O_y + \frac{1}{m} \sum_{i=1}^m p_i \sin \frac{2\pi i}{m} \\ P_z = O_z + \frac{1}{m} \sum_{i=1}^m p_i \end{cases}$$

Artero et al. [AdOL06] introduced axes arrangement for exhibition that keeps highly similar attributes close together, which may be achieved by computing information on the attributes similarity from the data set.

Coorprider and Burton [CB07] proposed an extension of star coordinates into three dimensions. The authors add a third dimension to traditional star coordinates, which allows for interaction in the third dimension, but it maintains the two-dimensional display. Three-dimensional star coordinates extend the traditional two-dimensional star coordinates in several ways:

- Stars distribute in a volume instead of a plane, giving users more space to exploit.
- Depth cues allow users to include more meaningful variables simultaneously in an analysis.
- Transformations are extended to three dimensions.
- System rotation is introduced as a powerful new transformation.

Shaik and Yeasin [SY06] presented an algorithm for an automated way of finding the best configuration when high-dimensional data points are projected into a 3D visual space. The best configuration of star coordinates is found among some random star coordinates configurations based on self-organizing maps clustering algorithm in visual space to measure quality of the star coordinates display. Shaik and Yeasin [SY07] proposed another algorithm for automatically finding the best configuration of star coordinates based on the minimization of a multidimensional scaling object function (stress function).

Chen and Liu [CL04] introduced VISTA mappings. The VISTA maps multidimensional data points into 2D visual space while providing the convenience of visual parameter adjustment:

$$\begin{cases} P_x = O_x + \frac{c}{m} \sum_{i=1}^m p_i \alpha_i \cos \theta_i, \\ P_y = O_y + \frac{c}{m} \sum_{i=1}^m p_i \alpha_i \sin \theta_i. \end{cases}$$

where $\alpha = (\alpha_1, \dots, \alpha_m)$ are the dimension adjustment parameters in $[-1, 1]$, angles $\theta = (\theta_1, \dots, \theta_m)$ are set to $\theta_i = \frac{2\pi i}{m}$ initially and can be adjusted, and c is the scaling of the radius of the display area. VISTA is an extension of traditional star coordinates that allows for more interactive exploration of multidimensional data. Also, Toeh and Ma [TM03] introduce starclass that allows interactive star coordinates for visual classification.

Sun et al. [STTX08] introduced advanced star coordinates that use the diameter instead of the radius as the dimensions, axis, such that data points in multidimensional space are mapped into visual space preserving attribute values with orthogonal distance from the visual point to the diameter. The diameters configuration strategy is based on correlations. The advanced star coordinates visualizes the clusters and structure of multidimensional data.

Dhillon et al. [DMS98, DMS02] proposed a method for projecting multidimensional data based on *class-preserving projection*. The authors presented an algorithm for finding the best two-dimensional plane that preserves inter-class distances. The mapping is a linear dimension reduction method, in which an optimized two-dimensional subspace is selected maintaining the distance between means of classes. In their paper, the authors did not discuss the relation with star coordinates.

2.1.4 Parallel coordinates

Parallel coordinates is one of the most popular visualization techniques for multidimensional multivariate data sets. Parallel coordinates are introduced by Inselberg [Ins85] and are developed for visualizing multidimensional geometry [ID90]. Parallel coordinates are based on a system of parallel coordinates, which includes a non-projective mapping between multidimensional and two-dimensional sets.

Parallel coordinates On the plane with Cartesian coordinates, and starting on the y -axis, m copies of the real line, labeled X_1, X_2, \dots, X_m , are placed equidistant and perpendicular to the x -axis. Typically, the X_i axis perpendicular to the x -axis lies at positions $i - 1$, for $i = 1, \dots, m$. They are the axes of the parallel coordinates system for the Euclidean m -dimensional space \mathbb{R}^m all having the same positive orientation as the y -axis. A point $P = (p_1, \dots, p_m)$ is represented by the polygonal line whose m vertices are at $(i - 1, p_i)$ on the X_i axes for $i = 1, \dots, m$, see Figure 2.2. In effect, a one-to-one correspondence between points in \mathbb{R}^m and planar polygonal lines with vertices on X_1, \dots, X_m is established.

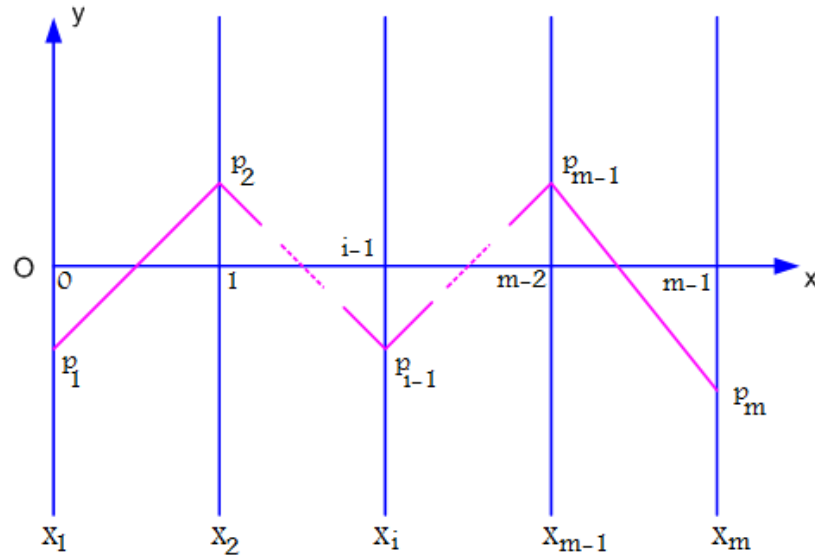


Figure 2.2: A polygonal line \bar{P} represents a point $P = (p_1, \dots, p_m)$.

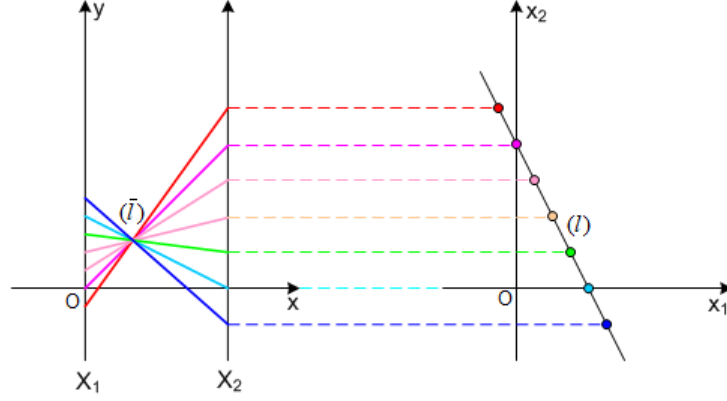


Figure 2.3: The dual *line* and *point* in parallel coordinates.

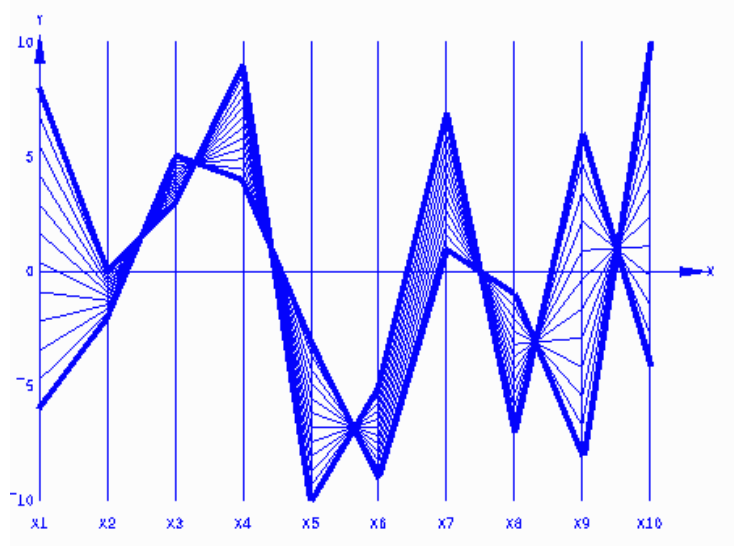


Figure 2.4: Parallel coordinates display an interval of a line in \mathbb{R}^{10} .

The fundamental duality We consider the X_1X_2 parallel coordinates as well as the Ox_1x_2 Cartesian coordinates that are shown in Figure 2.3. In the Cartesian coordinates Ox_1x_2 , we draw a line (l) that is described by the following equation:

$$(l) : x_2 = mx_1 + b.$$

Each point $(x_1, x_2 = mx_1 + b)$ lying on the line (l) in the Cartesian coordinates is displayed by a segment line with endpoints $(0, x_1)$ and $(1, x_2 = mx_1 + b)$ in parallel coordinates. Hence the points on (l) which are represented in parallel coordinates form an infinite family of lines. If $m \neq 1$, the family of lines has a common point:

$$(\bar{l}) : \left(\frac{1}{1-m}, \frac{b}{1-m} \right).$$

The point (\bar{l}) in parallel coordinates represents the line (l) in Cartesian coordinates. In the case $m = 1$, the family of lines has a common point at infinity with direction

$(1, b)$.

Each point in two-dimensional Cartesian coordinates is represented by a line in parallel coordinates and each point in parallel coordinates, which can be understood as a family of lines that intersect at this point, represents a line in Cartesian coordinates. This property is called a duality between *line and point*.

Multidimensional lines A line (l) in \mathbb{R}^m can be described by $m - 1$ linearly independent equations of the form:

$$(l) : \quad x_{i+1} = m_i x_i + b_i, i = 1, \dots, m - 1.$$

The line (l) is represented in parallel coordinates by $m - 1$ indexed points in the $X_i X_{i+1}$ parallel coordinates. In Figure 2.4 the points \bar{l} correspond to adjacent variables.

In the remaining of this subsection, we describe some applications of parallel coordinates for exploration data analysis.

Parallel coordinates in data analysis Wegman [Weg90] introduced a method to analyze data using parallel coordinates. In his paper, the author proposed two methods called *density plots* and *color histograms*. For density plots, the algorithm is based on Scott's notion of the Average Shifted Histogram (ASH) to visualize density plots with parallel coordinates. The author used contours to represent the two-dimensional density. Parallel coordinates density plots have the advantage of being graphical representations of data that are simultaneously high-dimensional and very large. In color histograms, the idea is to code the magnitude of an observation along a given axis by a color bin. The diagram is drawn by choosing an axis, and sorting the observations in ascending order. The author also introduced a permutation algorithm of the axes for pairwise comparisons.

Multiresolution view with parallel coordinates Fua et al. [FWR99] developed a multiresolutional view of the data via hierarchical clustering and use a variation on parallel coordinates to convey aggregation information for the results.

Novotny and Hauser [Nh06] introduced focus + context visualization in parallel coordinates. Each pair of adjacent axes representing a pair of dimensions, in a two-dimensional subspace is divided into $b \times b$ bins, which create a frequency-based and output-oriented representation of the original data.

Artero et al. [AdOL04] developed frequency and density-based visualizations. The basic idea of the algorithm is to create two-dimensional frequency histograms for each pair of adjacent attributes in parallel coordinates. A two-dimensional region between a pair of adjacent axes in parallel coordinates is divided into $w \times h$ bins, where w is the number of horizontal bins and h is the number of vertical bins. The value of frequency is stored in matrix $F = (F_{ij})_{w \times h}$. For each data point in multidimensional data sets, a line segment is drawn with the Bresenham algorithm, if the line segment goes through the (i, j) th bin, they add 1 to the value of F_{ij} . For the density plot, matrix frequencies $F = (F_{ij})_{w \times h}$ are linearly scaled into $[0, 255]$. For the frequency plot, they used a 3×3 averaging filter applied to the $F_{W \times H}$ matrix.

Johansson et al. [JLJC05, JLJC06] introduced a high-precision texture that can be used to reveal different types of cluster information. This visualization method can be used to analyze exclusive, overlapping, and hierarchical clusters. For displaying clusters in parallel coordinates, the authors used a transfer function on the intensity value which allows non-linear as well as user-defined mappings.

Generalization of parallel coordinates Perhaps the earliest multidimensional data visualization was introduced by Andrews [And72], in which each multidimensional data point $x = (x_1, \dots, x_m)$ is represented by a function of the form

$$f_x(t) = \frac{x_1}{\sqrt{2}} + x_2 \sin(t) + x_3 \cos(t) + \dots +$$

and this function is plotted on the range $[-\pi, \pi]$. Some useful properties of the Andrews' plots are preservation of means and distances. Theisel [The00] presented a free-form curve such that the space between two adjacent axes can be efficiently exploited to encode more information of the axes, which can help to detect correlations among more than two dimensions. Graham and Kennedy [GK03] used smooth curves to allow users to discern an individual path through the curves' nodes. Moustafa and Wegman [MW06] used a smooth plot between two adjacent axes. While in traditional parallel coordinates, a line segment can be understood as a linear interpolation, the authors introduced a new family of smooth functions using smooth interpolation. Zhou et al. [ZYQ⁺08] used curved lines to form visual bundles for clusters in parallel coordinates. The visual clustering is improved by adjusting the shape of the edges while keeping their relative order.

Dimension ordering in parallel coordinates Dimension ordering, spacing, and filtering can improve the parallel coordinates layout and facilitate data exploration. Ankerst et al. [ABK98] clustered data dimensions according to their similarity, then data dimensions are rearranged such that dimensions showing a similar behaviour are positioned next to each other. Yang et al. [YWR03] proposed a hierarchical approach to improve the interactivity of dimension reordering, spacing, and filtering. Peng et al. [PWR04] defined a visual clutter measure as the ratio of outlier points to the total data points. The optimized dimension order is then computed to minimize the proposed clutter measure.

Interacting with parallel coordinates Hauser et al. [HLD02] used angular brushing to pick out data subsets with specific trends between adjacent axes. Siirtola and Raiha [SR06] directly manipulated parallel coordinates by dynamically summarizing a set of polylines and interactively visualizing correlation between polyline subsets. These brushing and interactive techniques are considered very effective tools in exploring the structures within the clusters.

Integration with parallel coordinates Johansson et al. [JTJ04] used the self-organizing map in conjunction with parallel coordinates, in which clusters are represented instead of data points, which helps to see an overview and details in parallel

coordinates. Bertini et al. [BAS05] proposed the tight coupling between radviz and parallel coordinates called springview. In springview, the user can select a 2D area on the radviz representation getting the corresponding elements highlighted in the parallel coordinates cluttering. The color coding on the radviz (based on a 2D color-map to a rectangular board) is automatically computed, which allows for automatically clustering the parallel coordinates polylines, exploiting their similarity and their distances.

2.2 Cluster analysis

Clustering is the process of grouping the data into classes or clusters, that objects within a cluster have high similarity but are very dissimilar to objects in other clusters [HK06]. Clustering is a challenging field of research in which its potential applications pose their own special requirements. The following are typical requirements of clustering in data mining: *scalability*, *arbitrary shape*, *ability to deal with noisy data*, *high dimensionality*. In general, the major clustering methods can be classified into the following categories: partitioning methods, hierarchical methods, density-based methods, and grid-based methods.

Partitioning methods A partitioning method constructs k partitions of the data sets. It classifies the data set into k groups, which together satisfy the following requirements:

- each group must contain at least one object,
- each object must belong to exactly one group.

The most well-known and commonly used partitioning methods are k -means, and k -medoids [HK06].

Hierarchical methods A hierarchical method creates a hierarchical decomposition of a given set of data objects. A hierarchical method can be classified as being either *agglomerative (bottom-up)* or *divisive (top-down)*. The bottom-up approach starts with each object forming a separate group. It successively merges the objects or groups that are close to each other, until all the groups are merged into one, or until a termination condition holds. The top-down approach starts with all the objects in the same cluster. In each successive interaction, a cluster is split into two sub-clusters, until eventually each object is in one cluster, or until a termination condition holds.

Density-based methods Most partitioning methods cluster objects based on the distance between objects. Such methods can find only spherical-shaped clusters and encounter difficulty at discovering clusters of arbitrary shapes. Other clustering methods have been developed based on the notion of *density*. Their general idea is continued growing the given cluster as long as density is high in the neighborhood. Such a method can be used to filter out noise (outliers) and discover clusters of arbitrary shape.

Grid-based methods Grid-based methods quantize the object space into a finite number of cells forming a grid structure. All the clustering operations are performed on the grid structures. The main advantage of this approach is its low processing time, which is typically independent of the number of data objects and only dependent on the number of cells in each dimension of the quantized space.

2.2.1 Hierarchical clustering

A hierarchical algorithm divides a data set into a sequence of nested partitions. Hierarchical algorithms are divided into agglomerative hierarchical algorithms (bottom-up) and divisive hierarchical algorithms (top-down). Both agglomerative and divisive clustering methods organize data into hierarchical structures based on the similarity matrix.

Bottom-up The most common bottom-up hierarchical clustering algorithm is AGNES (AGglomerative NESTing). Initially, AGNES places each object into a cluster of its own. The clusters are merged step-by-step according to some criteria. Differences between methods arise because of the different ways of defining the distance (or similarity) between clusters.

Single linkage clustering: $d_{\min}(A, B) = \min_{a \in A, b \in B} d(a, b)$

Complete linkage clustering: $d_{\max}(A, B) = \max_{a \in A, b \in B} d(a, b)$

Average linkage clustering: $d_{ave}(A, B) = \frac{1}{|A||B|} \sum_{a \in A, b \in B} d(a, b)$

The AGNES algorithm can be summarized by the following procedure:

1. Start with n singleton clusters. Calculate the similarity matrix for the n clusters,
2. In the similarity matrix, find the minimal distance $d(C_i, C_j) = \min_{k, l} d(C_k, C_l)$, where $d(., .)$ is the distance function discussed above, and combine cluster C_i and C_j to form a new cluster C_{ij} ,
3. Update the similarity matrix by computing the similarity between the new cluster C_{ij} and the other clusters,
4. Repeat steps 2 and 3 until only one cluster remains.

When the algorithm uses $d_{\min}(A, B)$ it is also called a *nearest-neighbor hierarchical clustering algorithm*, and when the algorithm uses $d_{\max}(A, B)$ it is called a *farthest-neighbor hierarchical clustering algorithm*.

Top-down Divise clustering techniques are hierarchical in nature. The main difference with the bottom-up methods is that they proceed in the inverse order. At each step, a divisive method splits up a cluster into smaller ones, until finally all clusters contain only a single element. The divisive algorithm based on the same principle would start considering all divisions of the data set into two non-empty subsets, which amounts to $2^{n-1} - 1$ possibilities, where n is the size of the cluster to split. A complete enumeration approach is infeasible for large n .

Nevertheless, it is possible to construct divisive methods that do not consider all divisions. The most common top-down hierarchical clustering algorithm is DIANA (DIvisive ANALysis). Assuming a cluster X has n objects, the cluster X splits into two subsets X_r and X_l . Initially, we set X_r as an empty set and X_l as the entire set X . Next, we find the object x as

$$x = \arg \max_{y \in X_l} d(\{y\}, X_l \setminus \{y\}).$$

If the set X_r is empty, we move object x from X_l to X_r . If the set X_r is not empty, the object x is a candidate object. The candidate object x is moved to X_r if this object is more similar to X_r than similar to $X_l \setminus \{x\}$, i.e., if

$$d(\{x\}, X_r) < d(\{x\}, X_l \setminus \{x\}).$$

Eventually, the cluster splits into two subgroups X_r and X_l . The algorithm is recursively applied to each subgroup.

The DIANA algorithm is summarized in the following procedure:

1. *Initialization*: Start with C equal to $C_l = C$ and $C_r = \emptyset$ as an empty cluster.
2. *First iteration*:

- (a) For each data object $x \in C$ computes its average distance to all other objects:

$$d(x, C \setminus \{x\}) = \frac{1}{|C| - 1} \sum_{y \in C \setminus \{x\}} d(x, y)$$

- (b) Moving the data object x that achieves the maximum value $\max_{x \in C} d(x, C \setminus \{x\})$ to C_r .

3. *Remaining iteration loop*:

- (a) For each data object $x \in C$ computes the difference between the average distance to C_l and the average distance to C_r :

$$d(x, C_l \setminus \{x\}) - d(x, C_r) = \frac{1}{|C_l| - 1} \sum_{y \in C_l \setminus \{x\}} d(x, y) - \frac{1}{|C_r|} \sum_{y \in C_r} d(x, y)$$

- (b) If the maximum difference $\max_{x \in C} d(x, C_l \setminus \{x\}) - d(x, C_r)$ is greater than 0 then moving the data object x that achieves maximum difference to C_r and repeats the remaining iteration loop. Otherwise, the cluster C stops splitting.

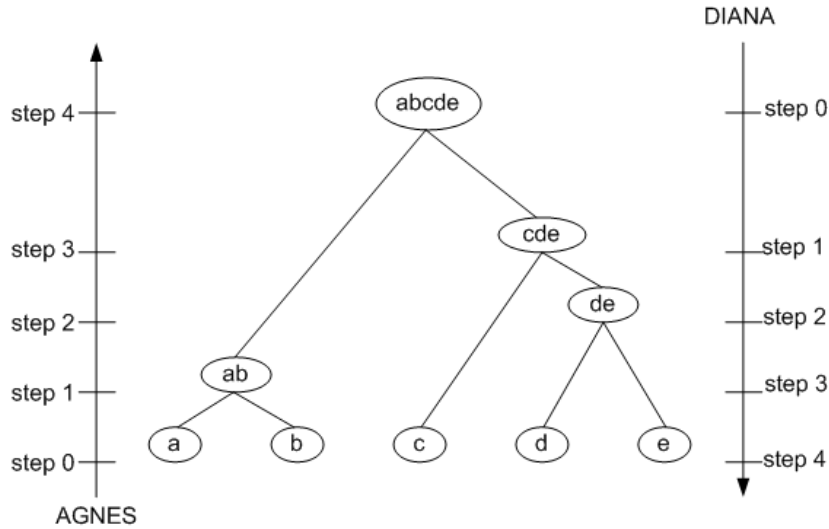


Figure 2.5: Agglomerative and divide hierarchical clustering on data objects $\{a, b, c, d, e\}$.

Figure 2.5 shows the application of AGNES algorithm and DIANA algorithm to a data set of five data objects, $\{a, b, c, d, e\}$. In AGNES, each object is placed into a cluster of its own. In the first step, two clusters $\{a\}$ and $\{b\}$ merge into the cluster $\{a, b\}$. In the second step, two clusters $\{d\}$ and $\{e\}$ merge into the cluster $\{d, e\}$ and this cluster is combined with the cluster $\{c\}$ to the cluster $\{c, d, e\}$ in the third step. Finally, two clusters $\{a, b\}$ and $\{c, d, e\}$ merge into one cluster $\{a, b, c, d, e\}$. In DIANA, all of objects are used to form one initial cluster $\{a, b, c, d, e\}$. The first step of DIANA algorithm splits this cluster into two clusters $\{a, b\}$ and $\{c, d, e\}$. The cluster $\{a, b\}$ is splitted into the two clusters $\{a\}$ and $\{b\}$. The cluster $\{c, d, e\}$ is splitted into the two clusters $\{c\}$ and $\{d, e\}$ and the cluster $\{d, e\}$ is splitted into the two clusters $\{d\}$ and $\{e\}$.

Recent advances The common criticism for classical hierarchical clustering algorithms is high computational complexity, which is at least $O(n^2)$. This high computational burden limits their application to large-scale data sets. In order to address this problem and other disadvantages, some new hierarchical clustering algorithms have been proposed, such as BIRCH [ZRL96], CURE [GRS98], and CHAMELEON [KHK99].

BIRCH uses a hierarchical data structure called CF-tree for partitioning the incoming data points in an incremental and dynamic way. The CF-tree is a height-balanced tree, which stores the clustering features and is based on two parameters: branching factor B and threshold T . The branching factor B refers to the maximum number of children per internal node and the threshold T refers to the maximum radius of the cluster (the average distance from points in the cluster to the centroid) or the maximum diameter of the cluster (the average pairwise distance within the cluster). A leaf node of the CF-tree

contains at most L data points.

A CF-tree is built as the data is scanned. When reading a new data point the CF-tree is traversed starting from the root, it recursively descends the CF-tree by choosing the closest child node to the new data point according to a distance metric between two clusters (centroid or average distance).

When the closest leaf node for the current data point is finally identified, a test is performed to see whether adding the data item to the candidate cluster if without violating the threshold conditions or the leaf node is splitted. The leaf node splitting is done by choosing the farthest pair of data points, and partitioning the remaining data points based on the closest criteria.

BIRCH applies an agglomerative hierarchical clustering algorithm to cluster the leaf nodes of the CF-tree, which removes sparse clusters as outliers and merges dense clusters into larger ones.

BIRCH may not work well when clusters are not spherical because it uses the concept of radius or diameter to control the boundary of a cluster.

CURE Instead of using a single centroid to represent a cluster, a constant number of representative points is chosen to represent a cluster. The number of points chosen is governed by a parameter c .

The similarity between two clusters is measured by the minimum distance between the representative points of these clusters. Like AGNES algorithm, at each step the closest pair of clusters is merged to form a new cluster. Representative points of the new cluster are computed by iteration: the farthest point from the centroid of the new cluster is chosen as the first scattered point and in each iteration, a point from the new cluster is chosen by the farthest point from the previously chosen scattered points. The points are shrunk toward the centroid by a fraction parameter α .

Unlike centroid/medoid based methods, CURE is capable of finding clusterings of arbitrary shapes and sizes, as it represents each cluster via multiple representative points. Shrinking the representative points toward the centroid helps CURE in avoiding the problem of a noise present in the single link method. However, CURE cannot be applied directly to large data sets.

CHAMELEON finds clusters in data sets by using a two-phase algorithm. In the first step, it generates a k -nearest neighbor graph that contains links only between a point and its k -nearest neighbors. In the second step, CHAMELEON uses a graph-partitioning algorithm to cluster the data points into a large number of relatively small sub-clusters. The similarity between two clusters is determined according to their *relative interconnectivity* and *relative closeness* [GRS98]. During the second phase, it uses an agglomerative hierarchical clustering algorithm to find the genuine clusters by repeatedly combining together these sub-clusters. No cluster may contain less than a user specific number of instances.

2.2.2 Hierarchical density clusters

High-density clusters are defined on a population with a density function $p(x)$ in m dimensions to be the maximal connected set of form $\{x : p(x) \geq \lambda\}$ at a level set λ [Har75].

Wong [Won82] presented a hybrid clustering to construct high density clusters. First, the data set is partitioned into k clusters by a partitioning method (k -means). Second, a single linkage clustering technique is applied to the distance matrix based on the mean of the k clusters.

Wong and Lane [WL83] developed the k th-nearest procedure to build a high density cluster. First, a density function of the data set is estimated based on the k nearest neighbors. The distance between two objects is defined as the average of the inverse of the density distribution:

$$d(x, y) = \frac{1}{2} \left(\frac{1}{p(x)} + \frac{1}{p(y)} \right).$$

The single linkage clustering is applied to the distance matrix D to obtain the sample tree of high-density clusters.

Stuetzle [Stu03] constructed a high density cluster based on analyzing the minimal spanning tree (MST). The author defined the *runt size* for an edge e of the MST: breaking all MST edges that have length greater than the length of e into subtrees, the runt size of the edge e is the smallest number of nodes of those subtrees. The idea of runt pruning considers a split of the MST into two connected components to be significant if both children contain a sufficiently large number of observations. Each node N of a cluster represents a subtree $T(N)$ of the MST and an associated with a density level $\lambda(N)$. The root node of the cluster tree represents the MST of the entire data and density level $\lambda = 0$. The cluster tree is recursively constructed. For each node N , the longest edge e in $MST(N)$ with the runt size larger than a threshold parameter is chosen. If there is no such edge then N is a leaf node of the cluster tree. Otherwise, breaking all the edges of $MST(N)$ with length greater or equal the length of e results in a subgraph of $MST(N)$ and these subgraphs are children of the node N associated with density level $\lambda = \frac{2^m}{nV||e||^m}$, where n is the size of the data set, m is the dimension of the data set, V is the volume of the unit sphere in m dimensional space, and $||e||$ is the length of the edge.

Ester et al. [EKSX96] proposed DBSCAN (Density Based Spatial Clustering of Applications with Noise) to discover arbitrarily shaped clusters. DBSCAN requires the setting of two parameters: *Eps* to define density for data points and *MinPts* to define core points. A data point x is called a core point if there are at least *MinPts* data points that fall inside the ball $B(x, Eps)$. A cluster is a connected region that can be represented by a union set of balls with centers at the core points and radius *Eps*. Border points are data points that belong to clusters and are not core points. The clustering result is sensitive to the choice of the parameter *Eps*. DBSCAN may not handle data sets that contain clusters with different densities. Ankerst et al. [ABKS99] introduced the OPTICS (Ordering Points To Identify the Clustering Structure) algorithm that detects meaningful clusters in data with varying

densities. To overcome the limitation of DBSCAN, the authors used two concepts: core-distance and reachability-distance. The core-distance of a core point x is the smallest distance Eps' such that x is a core point with respect to Eps' . Otherwise, the core-distance is undefined. The reachability-distance of a data point x with respect to a core point y is the smallest distance such that x belongs to the $MinPts$ nearest neighbors of y . Thus, the reachability-distance of x with respect to y is the maximum of the core-distance of y and the distance between x and y . If y is not a core point, the reachability-distance of x with respect to y is undefined. OPTICS creates an ordering of a data set by sorting with respect to the core-distance and a suitable reachability-distance for each data point. Clusters are identified in the OPTICS approach by a reachability-distance plot.

Hinneburg and Keim [HK98, HK03] proposed a general approach to clustering based on kernel density estimation called DENCLUE (DENsity-based CLUstEring). The density estimation is defined as the sum of kernel functions of all data points:

$$f(x) = \frac{1}{nh^m} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right),$$

where $\{x_i \in \mathbb{R}^m : 1 \leq i \leq n\}$ are n data points in an m -dimensional space, $K(x)$ is a kernel function, and h is a smoothness parameter.

The DENCLUE algorithm works in two steps. The first step efficiently approximates the density function. The data space is divided into hypercubes with an edge length of $2h$, and only populated hypercubes are determined. Two hypercubes c_1 and c_2 are said to be connected if $D(\text{mean}(c_1), \text{mean}(c_2)) \leq 4h$, where $\text{mean}(c)$ is the barycenter of all data points inside hypercube c and $D(\cdot, \cdot)$ is a distance between two m -dimensional points. The local density estimation $\hat{f}(x)$ is

$$\hat{f}(x) = \frac{1}{nh^m} \sum_{y \in \text{near}(x)} K\left(\frac{x - y}{h}\right),$$

where $y \in \text{near}(x)$ if $D(x, y) \leq 4h$. The second step is the clustering step using a hill-climbing procedure. It is guided by the gradient $\nabla \hat{f}(x)$ of the local density function. The density attractor for a point x is computed iteratively as

$$x^0 = x, x^{i+1} = x^i + \delta \frac{\nabla \hat{f}(x^i)}{\|\nabla \hat{f}(x^i)\|}.$$

The δ is a parameter of the hill-climbing procedure that controls the speed of convergence. The hill-climbing procedure stops when $\hat{f}(x^{k+1}) < \hat{f}(x^k)$ and takes $x^* = x^k$ as a new density attractor. If $\hat{f}(x^*) > \xi$ (ξ a threshold density parameter), then x is assigned to the cluster belonging to x^* .

It is interesting to point out that DENCLUE provides a generalization of different cluster paradigms: partition-based, density-based single linkage, and hierarchical clustering. In hierarchical approaches, the authors propose using different smoothness level to generate a hierarchy of clusters. When starting DENCLUE with a small value for h (h_{\min}), one may obtain N clusters. With increasing h , certain point density attractors start to merge and one obtains the next level of the hierarchy. If one

further increases h , more and more density attractors merge and, finally, only one density attractor representing the root of the hierarchy is left.

2.3 Visualization of hierarchical clustering results

A hierarchical clustering can be represented by either a picture or a list of abstract symbols. A picture of hierarchical clustering results is much easier for humans to interpret. A list of abstract symbols of a hierarchical clustering may be used internally to improve the performance of the algorithm. A hierarchical clustering is generally represented by a tree diagram.

2.3.1 Visualizing hierarchical structures

The dendrogram is a graphical representation of the results of hierarchical cluster analysis. This is a tree-like plot where each step of the hierarchical clustering is represented as a fusion of two branches of the tree into a single one. The branches represent clusters obtained in each step of the hierarchical clustering. Figure 2.6 shows a dendrogram of a data set of five data objects, $\{a, b, c, d, e\}$.

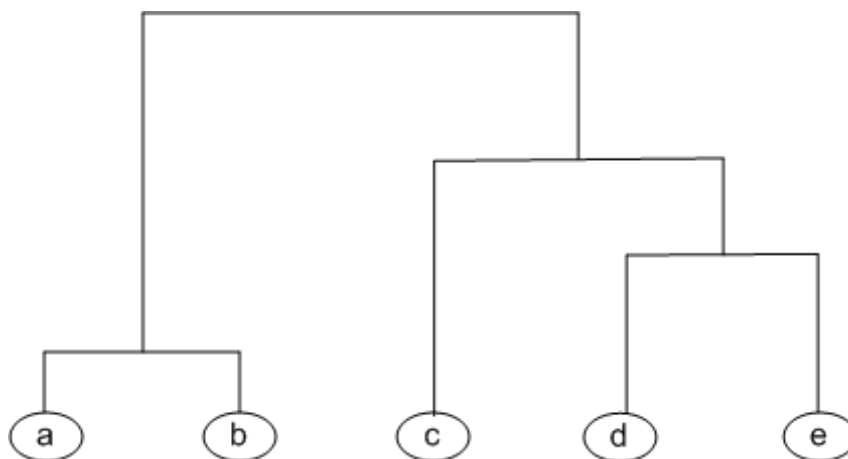


Figure 2.6: Dendrogram representation of hierarchical clustering results.

The classical hierarchical view is based on the algorithms developed by Reingold and Tilford [RT81]. The algorithm computes independently the relative positions of subtrees and then joins them in a larger tree by placing these subtrees as close together as possible. It can be adapted to produce top-down as well as left-to-right tree layouts, and can also be set to output grid-like positioning. The algorithm is simple, fast, and predictable.

Treemaps were introduced by Johnson and Shneiderman [JS91]. In the treemaps, the hierarchical structure is mapped to nested rectangles. Treemap is constructed by recursive subdivision, i.e., a node is divided into some rectangles based on the children's size of this node. The direction of subdivision alternates, a rectangle is subdivided in one direction (for instance, horizontally), and for the next level this

direction alternates. Treemaps provide a compact visual representation of complex hierarchical data.

Another technique for visualizing hierarchical data sets is the so-called Inter-Ring [YWR02], which displays nodes of a hierarchy by ring segments. All child nodes are arranged on concentric circles; the further they are away from the root node, the deeper their level within the hierarchy. For each node, all respective ancestor nodes can be found in between the ring segment representing the considered node, and the center of the InterRing.

A radial drawing [TBET99] is a variation of a layered drawing where the root of the tree is placed at the origin and layers are concentric circles centered at the origin. In radial drawings, a subtree is usually drawn within an annulus wedge. Teoh and Ma [TM02] introduced a technique for visualizing large hierarchies: RINGS, a node and its children are placed in a circle. FOXI [CK06] is an approach to achieve the ability to display infinite hierarchy size in a limited display area.

Another research focused on using planar methods to display hierarchies, as any tree can be drawn in $2D$ without intersecting edges. One possibility to display a hierarchy is the hyperbolic tree [LRP95]. This visualization technique enables focus + context visualizations by taking advantage of a hyperbolic projection which scales nodes according to their proximity to the focal point.

Information visualization has contributed with helpful ideas for displaying hierarchies. One of them is the conetree [RMC91], a $3D$ visualization that orders child nodes on a circle below or next to their parent node. When the links between parent and child nodes are drawn, cone-like structures appear.

Information cube [RG93] uses semi-transparent nested boxes or nested cubes to represent the hierarchical information. It represents the parent-child relationships by recursively placing child cubes inside their parent cubes. The outermost cube is the top level of data. All the cubes are transparent so that the nested subtree can be viewed inside the cube.

2.3.2 Visualizing hierarchical density clusters

Most visualization systems rely upon a two-dimensional representation and use a fixed layout algorithm. The use of a three-dimensional representation through which users can navigate provides a much richer visualization. The Narcissus system was introduced by Hendley et al. [HDWB95]. The spatial layout of objects is based upon physical systems with rules defining forces that act between the objects in a $3D$ visual space. These forces cause the objects to move in space. Objects migrate through space so that they are spatially close to those objects with which they are semantically related. High-similarity objects merge into one compound object. The compound objects are formed by placing a translucent surface around the cluster so that from a distance it appears as one distinctive object, but as it is approached, the internal structure becomes more apparent and the user can be smoothly moved from a high-level view to one in which all the details are available.

Sprenger et al. [SBG00] introduced the h-blob system, which groups and visualizes cluster hierarchies at multiple levels of detail. The h-blob includes two steps, a cluster tree is computed making use of an edge collapse clustering and visualizes the

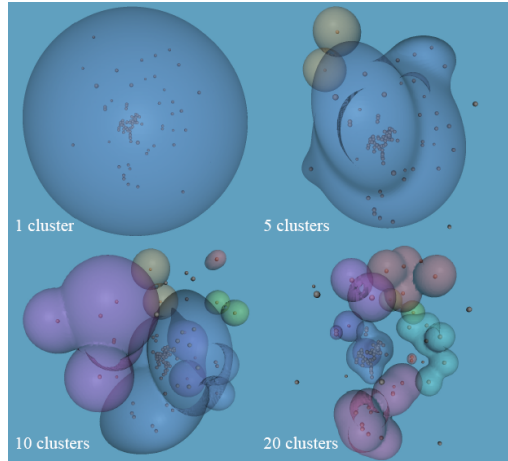


Figure 2.7: Cluster hierarchies are shown for 1, 5, 10 and 20 clusters.

clusters by computing a hierarchy of implicit surfaces. The most significant feature of hierarchical blobs is not only to provide the overview of the entire data set, but also to give a detailed visual representation of clusters. The high visual complexity of the two stages of blob graph formation makes them unsuitable for being applied in cluster visualization of very large data sets. Another limitation of h-blob is that the hierarchical clustering is executed in visual space, i.e., it does not cluster the original data set. Figure 2.7 shows nested blobs with many levels of detail.

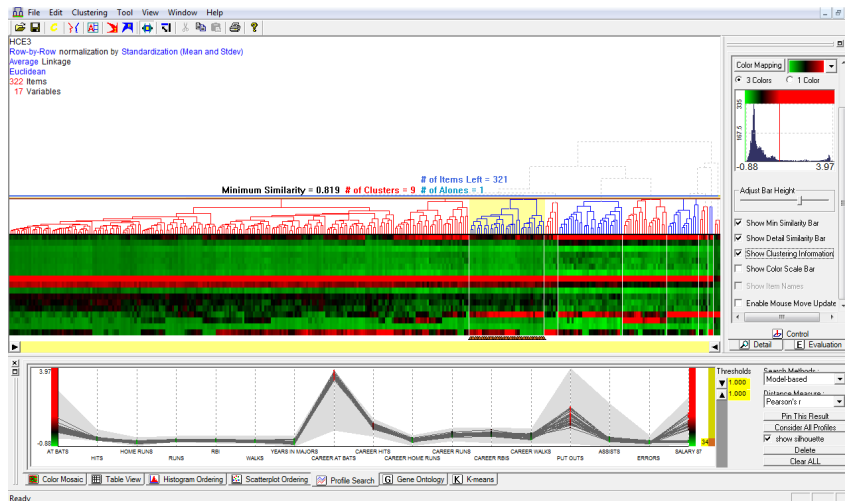


Figure 2.8: Hierarchical clustering explorer for interactive exploration of multidimensional data.

Seo and Shneiderman [SS02, SS05] introduced a Hierarchical Clustering Exploration (HCE), which integrates four interactive features: overview of the entire data set, dynamic query controls, coordinated displays that enforce a bidirectional link from the overview mosaic to two-dimensional scattergrams, and cluster comparisons.

HCE uses agglomerative hierarchical clustering with different kinds of distances, the hierarchical clustering is displayed with a traditional dendrogram. HCE uses a similarity bar that controls a similarity value and the hierarchical clustering cuts off all similarity edges that are smaller than the value of the similarity bar indication and the hierarchical clustering is divided into some clusters (cut the dendrogram and derive clusters). Figure 2.8 displays HCE for interactive exploration of multidimensional data. Nine clusters are obtained by cutting the dendrogram at the similarity value 0.819. HCE is a good method to display hierarchical clustering as it provides an intuitive and easily understood visual presentation. However, HCE has some limitation: clusters are not automatically identified from hierarchical clustering, i.e., clusters are dependent on the value of the similarity bar and the dendrogram is very complex for large data set.

Chapter 3

Hierarchical density clusters

The goal of this chapter is to present a criterion for nonparametric density estimation and a hierarchical density cluster construction based on nonparametric density estimation. We propose two variants of efficient nonparametric density estimation. The first variant uses a traditional multivariate histogram method. The second estimates the density function with a kernel method. Corresponding with the two methods of density estimation, we propose two algorithms to create hierarchical density clusters using a bottom-up and a top-down strategy, respectively. The main advantage of our approach to build hierarchical density clusters is the direct identification of clusters without any threshold parameter of density level sets.

3.1 Nonparametric density estimation

3.1.1 Nonparametric estimation criteria

In the case of parametric density estimation, a parametric density family $p(x|\theta)$ is given (for example, the two-parameter normal distribution family $N(\mu, \sigma^2)$ where $\theta = (\mu, \sigma^2)$). The objective is to obtain the best estimator $\hat{\theta}$ of θ . In the case of nonparametric density estimation, the emphasis is directly on obtaining a good estimate $\hat{p}(x)$ of the entire density function $p(x)$. In this section, we present some estimation criteria for nonparametric density estimation.

Unbiasedness Let X_1, \dots, X_n be independent and identically distributed random vector variables in \mathbb{R}^m with an unknown continuous density $p(x)$, i.e.,

$$p(x) \geq 0, \int_{\mathbb{R}^m} p(x) dx = 1.$$

The problem of nonparametric density estimation is to estimate $p(x)$ based on observations X_1, \dots, X_n . Rosenblatt [Ros56] shows that, if the estimator $\hat{p}(x) = \hat{p}(x; X_1, \dots, X_n)$ is an estimation of the density function $p(x)$, the estimator $\hat{p}(x)$ is not unbiased, i.e.,

$$E[\hat{p}(x)] \neq p(x),$$

where $E[\hat{p}(x)]$ is the expectation over the random variables X_1, \dots, X_n . Although the estimator of the density function is not unbiased, the estimation needs to be asymptotically unbiased, i.e.,

$$\lim_{n \rightarrow \infty} E[\hat{p}(x)] = p(x).$$

Consistency There are other measures of discrepancy between the theoretical density function and its estimate. The mean squared error (MSE) is defined by

$$MSE(\hat{p}(x)) = E[\hat{p}(x) - p(x)]^2.$$

The above equation can be written as

$$\begin{aligned} MSE(\hat{p}(x)) &= E[\hat{p}(x) - p(x)]^2 \\ &= E\left[\left(\hat{p}(x) - E[\hat{p}(x)]\right) + \left(E[\hat{p}(x)] - p(x)\right)\right]^2 \\ &= E\left(\hat{p}(x) - E[\hat{p}(x)]\right)^2 + \left(E[\hat{p}(x)] - p(x)\right)^2 \\ &= V[\hat{p}(x)] + \left(B[\hat{p}(x)]\right)^2, \end{aligned}$$

where $V[\hat{p}(x)] = E[\hat{p}(x) - E\hat{p}(x)]^2$ is the variance and $B[\hat{p}(x)] = E[\hat{p}(x)] - p(x)$ is the bias.

A global measure of accuracy is given by the integrated squared error (ISE)

$$ISE(\hat{p}) = \int_{\mathbb{R}^m} [\hat{p}(x) - p(x)]^2 dx,$$

and by the mean integrated squared error (MISE) which represents an average over all possible data sets

$$MISE(\hat{p}) = E\left(\int_{\mathbb{R}^m} [\hat{p}(x) - p(x)]^2 dx\right) = \int_{\mathbb{R}^m} V[\hat{p}(x)] dx + \int_{\mathbb{R}^m} \left(B[\hat{p}(x)]\right)^2 dx. \quad (3.1)$$

We use the above criterion to find an estimate for the density function that minimizes $MISE(\hat{p})$ among some special class of estimator density functions.

3.1.2 Multivariate histogram density estimation

The histogram method is perhaps the oldest method of nonparametric density estimation. It is a classical method by which a probability density is constructed from a set of observations.

In one dimension, the real line \mathbb{R} is partitioned into a number of equally-sized cells and an estimator of a univariate density function at a point x is taken to be

$$\hat{p}(x) = \frac{n_i}{nh},$$

where n_i is the number of samples in the cell of width h that straddles the point x and n is the size of the data set. Figure 3.1 displays an example of a univariate histogram. Similarly, in two dimensions the plane \mathbb{R}^2 is partitioned into a number of equally-sized cells. Let the length and the width of the cells be denoted by h_1 and h_2 . An estimator of a bivariate density function at any point $x = (x_1, x_2)$ inside a cell B_i is defined by

$$\hat{p}(x_1, x_2) = \frac{n_i}{nh_1h_2},$$

where n_i is the number of points falling inside the cell B_i . Figure 3.2 shows a bivariate histogram.

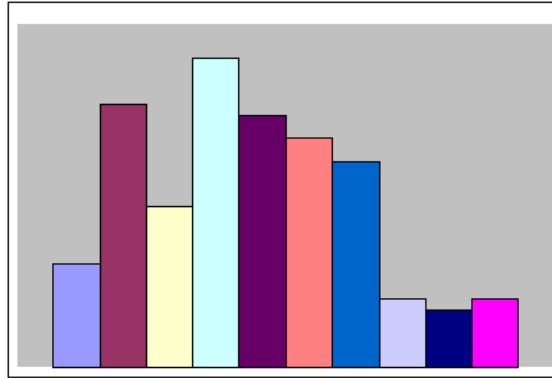


Figure 3.1: Univariate nonparametric density estimation using the histogram method.

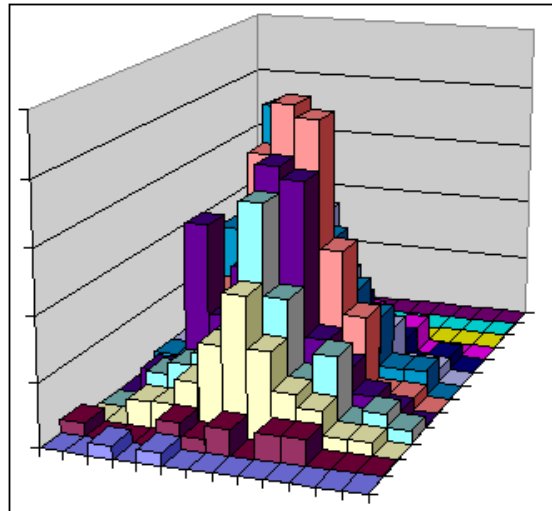


Figure 3.2: Bivariate nonparametric density estimation using the histogram method.

We can easily extend this method to multivariate histograms. Let $\{X_1, \dots, X_n\}$ be a random vector of variable observations from an unknown density function $p(x)$

in \mathbb{R}^m . The bounding region of the data set is divided into hyperrectangles of size $h_1 \times \dots \times h_m$, i.e., the length of each hyperrectangle of the j th dimension is h_j for all $j = 1, \dots, m$. If n_i observations fall into a hyperrectangular cell B_i , then the multivariate histogram estimator has the form

$$\hat{p}(x) = \frac{n_i}{nh_1 \dots h_m}, \quad (3.2)$$

for any point $x = (x_1, \dots, x_m)$ belonging to the cell B_i .

The best estimation of the bandwidth h_1, \dots, h_m such that the *MISE* is minimized in Equation (3.1) is a critical problem in nonparametric density estimation. We start with considering the first term of $MISE(\hat{p})$, the variance of the multivariate histogram.

Let p_k be the probability of the cell B_k , i.e.,

$$p_k = \int_{B_k} p(x) dx.$$

Thus, the distribution of n_k , the number of observations in the cell B_k , has the binomial distribution $B(n, p_k)$ and we have

$$V(n_k) = np_k(1 - p_k).$$

The variance of $\hat{p}(x)$ is constant over the cell B_k and it is given by

$$V[\hat{p}(x)] = \frac{V(n_k)}{(nh_1 \dots h_m)^2} = \frac{np_k(1 - p_k)}{(nh_1 \dots h_m)^2},$$

for all $x \in B_k$. Integrating the variance over the cell B_k and using $\sum_{B_k} p_k = 1$, we have

$$\int_{\mathbb{R}^m} V[\hat{p}(x)] dx = h_1 \dots h_m \sum_{B_k} \frac{np_k(1 - p_k)}{(nh_1 \dots h_m)^2} = \frac{1}{nh_1 \dots h_m} + O\left(\frac{1}{nh_1 \dots h_m}\right). \quad (3.3)$$

Next, we consider the second term in Equation (3.1), the bias of the multivariate histogram. To do so, we estimate the probability p_k by using the standard multivariate Riemannian integration approximation,

$$p_k = \int_{B_k} p(x) dx = h_1 \dots h_m p(c_k) + O(h^{m+2}),$$

where $h = \min_{1 \leq j \leq m} h_j$ and c_k is the center of the cell B_k . Thus, the expression $B[\hat{p}(x)]$ over the cell B_k is given by

$$B[\hat{p}(x)] = E[\hat{p}(x)] - p(x) = p(c_k) - p(x) + O(h^2)$$

and

$$B[\hat{p}(x)] = -(x - c_k)^t \nabla p(c_k) + O(h^2).$$

The integrated squared bias over the cell B_k is

$$\int_{B_k} (B[\hat{p}(x)])^2 dx = \frac{1}{12} \sum_{j=1}^m h_j^2 \int_{B_k} p_j^2(x) dx + O(h^4),$$

where $p_j(x) = \frac{\partial p(x)}{\partial x_j}$, $j = 1, \dots, m$. Summing over all cells yields

$$\int_{\mathbb{R}^m} (B[\hat{p}(x)])^2 dx = \frac{1}{12} \sum_{j=1}^m h_j^2 \int_{\mathbb{R}^m} p_j^2(x) dx + O(h^4). \quad (3.4)$$

The $MISE(\hat{p})$ of the multivariate histogram for density estimation is obtained from Equation (3.3) and Equation (3.4):

$$MISE[\hat{p}] = \frac{1}{nh_1 \dots h_m} + \frac{1}{12} \sum_{j=1}^m h_j^2 \int_{\mathbb{R}^m} p_j^2(x) dx + O\left(\frac{1}{nh_1 \dots h_m}\right) + O(h^4).$$

The asymptotic MISE approximation (AMISE) (the leading terms in the expression of MISE) is given by

$$AMISE(h_1, \dots, h_m) = \frac{1}{nh_1 \dots h_m} + \frac{1}{12} \sum_{j=1}^m h_j^2 \int_{\mathbb{R}^m} p_j^2(x) dx. \quad (3.5)$$

The cell widths that minimize AMISE [Sco92] are thus

$$h_j = \left(6 \prod_{j=1}^m \|p_j\|_2\right)^{\frac{1}{m+2}} \|p_j\|_2^{-1} n^{-\frac{1}{m+2}}, j = 1, \dots, m,$$

where $\|p_j\|_2^2 = \int_{\mathbb{R}^m} p_j^2(x) dx$, $j = 1, \dots, m$.

If the reference distribution is multivariate normal and the different variables being independent with possibly different standard deviations σ_j . Scott [Sco92] shows the optimal choice for the cell widths as

$$h_j = 3.5\sigma_j n^{-\frac{1}{m+2}}, j = 1, \dots, m.$$

3.1.3 Multivariate kernel density estimation

The multivariate kernel density estimation is defined as

$$\hat{p}(x) = \frac{1}{nh^m} \sum_{i=1}^n K\left(\frac{x - X_i}{h}\right), \quad (3.6)$$

where $K(x)$ is a multivariate kernel density function in \mathbb{R}^m such that

$$K(x) \geq 0, \int_{\mathbb{R}^m} K(x) dx = 1, \int_{\mathbb{R}^m} x K(x) dx = 0, \text{ and } \int_{\mathbb{R}^m} x^t x K(x) dx = \mu_2(K) I_m$$

($\mu_2(K)$ is a constant and I_m is the identity matrix), and h is the bandwidth or smoothing parameter. A common approach for multivariate density function estimation is to construct the multivariate kernel density function as a sum of the product of univariate kernel density functions

$$\hat{p}(x) = \frac{1}{nh_1 \dots h_m} \sum_{i=1}^n \prod_{j=1}^m K_j\left(\frac{x_j - X_{ij}}{h_j}\right), \quad (3.7)$$

where h_1, \dots, h_m are bandwidth parameters and K_1, \dots, K_m are univariate kernel density functions. Usually, K_1, \dots, K_m are taken to be of the same form. In general, the multivariate kernel density estimation can be written as:

$$\hat{p}(x) = \frac{1}{n \det(H)} \sum_{i=1}^n K(H^{-1}(x - X_i)) = \frac{1}{n} \sum_{i=1}^n K_H(x - X_i), \quad (3.8)$$

where H is the parameter matrix and $K_H(x) = \frac{1}{\det(H)} K(H^{-1}x)$.

One of the problems with multivariate kernel density estimation is the choice of the smoothing parameters. Let's consider the formulation in Equation (3.6). If h is too small, the density estimator is a collection of n sharp peaks, positioned at the sample points. If h is too large, the density estimation is smoothed and structure in the probability density estimation is lost. An optimal choice of h depends on several factors. Obviously, it depends on the data, i.e., on the number of data points and their distribution. It also depends on the choice of the kernel function and on the optimal criterion used for its estimation.

We evaluate $MISE(\hat{p})$ given in Equation (3.1) for the multivariate kernel density estimation in Equation (3.8). For the bias term, we derive

$$\begin{aligned} E[\hat{p}(x)] &= \int_{\mathbb{R}^m} K_H(x - u) p(u) du \\ &= \int_{\mathbb{R}^m} K(w) p(x + Hw) dw \\ &\approx \int_{\mathbb{R}^m} K(w) \left[p(x) + w^t H^t \nabla p(x) + \frac{1}{2} w^t H^t \nabla^2 p(x) H w \right] dw \\ &\approx p(x) + \frac{1}{2} \mu_2(K) \text{tr}(H^t \nabla^2 p(x) H) \end{aligned}$$

where $\nabla p(x)$ is the gradient vector of $p(x)$, $\nabla^2 p(x)$ is the Hessian matrix of $p(x)$, and $\text{tr}(\cdot)$ denotes the trace of a matrix. Therefore,

$$B[\hat{p}(x)] \approx \frac{1}{2} \mu_2(K) \text{tr}(H^t \nabla^2 p(x) H). \quad (3.9)$$

For the variance term, we obtain

$$V[\hat{p}(x)] = \frac{1}{n} \int_{\mathbb{R}^m} K_H^2(x - u) p(u) du - \frac{1}{n} \left(E[\hat{p}(x)] \right)^2$$

$$\begin{aligned}
&\approx \int_{\mathbb{R}^m} \frac{1}{n \det(H)} K^2(w) p(x + Hw) dw \\
&\approx \int_{\mathbb{R}^m} \frac{1}{n \det(H)} K^2(w) \left[p(x) + w^t H^t \nabla p(x) \right] dw \\
&\approx \frac{p(x)}{n \det(H)} \|K\|_2^2.
\end{aligned}$$

where $\|K\|_2^2 = \int_{\mathbb{R}^m} K^2(x) dx$. Hence, we obtain the AMISE [Sco92]

$$AMISE[H] = \frac{1}{4} \mu_2^2(K) \int_{\mathbb{R}^m} \left[\text{tr} \left(H^t \nabla^2 p(x) H \right) \right]^2 dx + \frac{1}{n \det(H)} \|K\|_2^2. \quad (3.10)$$

Case $H = hI_m$: The AMISE can be written as

$$AMISE(h) = \frac{1}{4} \mu_2^2(K) h^4 \int_{\mathbb{R}^m} \left[\text{tr} \left(\nabla^2 p(x) \right) \right]^2 dx + \frac{1}{nh^m} \|K\|_2^2. \quad (3.11)$$

The bandwidth parameter h that minimizes $AMISE(h)$ is thus $h = O(n^{-\frac{1}{m+4}})$. We have to find the value of h that minimizes the mean integrated squared error between the density and its approximation. For a radially symmetric normal kernel, Silverman [Sil86] suggested

$$h = \sigma \left(\frac{4}{m+2} \right)^{\frac{1}{m+4}} n^{-\frac{1}{m+4}} \quad (3.12)$$

where a choice for σ is

$$\sigma^2 = \frac{1}{m} \sum_{j=1}^m s_{jj}$$

and s_{jj} are the diagonal elements of the sample covariance matrix.

Case $H = \text{diag}[h_1, \dots, h_m]$: The AMISE can be written as,

$$AMISE(h_1, \dots, h_m) = \frac{1}{4} \mu_2^2(K) \sum_{j=1}^m h_j^4 \int_{\mathbb{R}^m} \left(\nabla^2 p(x) \right)_{jj} dx + \frac{1}{nh_1 \dots h_m} \|K\|_2^2. \quad (3.13)$$

In the simplest case, we consider the density function $p(x)$, that have the normal distribution with the diagonal matrix covariance $\Sigma = \text{diag}(\sigma_1^2, \dots, \sigma_m^2)$. Scott [Sco92] suggested

$$h_j = n^{-\frac{1}{m+4}} \sigma_j, j = 1, \dots, m. \quad (3.14)$$

3.2 Hierarchical density clusters

Given a multivariate density function $p(x)$ of a random vector of variables in \mathbb{R}^m . In density-based clustering, clusters can be understood as regions of high density of data points in the multidimensional space that are separated from other regions by regions of low density, i.e., the density of data points within a cluster is higher than the density of data points around the cluster. In statistics, a mode of the density function $p(x)$ is a point x where $p(x)$ has a local maximum. Thus, a mode of a given distribution is more dense than its surrounding area. A cluster which contains only one mode of the density function is called a *homogeneous cluster* and a cluster that contains more than one mode of the density function is called a *heterogeneous cluster*. One heterogeneous cluster can contain heterogeneous clusters and homogeneous clusters. Therefore, the homogeneous clusters and the heterogeneous clusters form hierarchical density clusters. The goal is to find these clusters and the hierarchical structure. We can understand the hierarchical structure of density clusters of the multivariate density function through level sets.

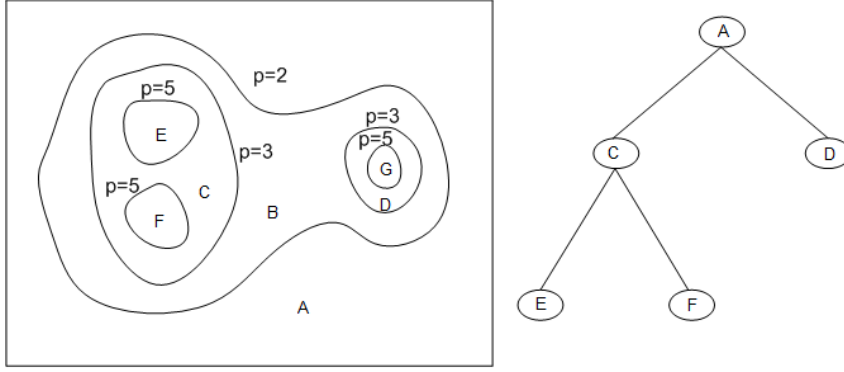


Figure 3.3: Hierarchical density clusters.

Let $0 < \lambda < \sup_{x \in \mathbb{R}^m} p(x)$. We are considering regions of the sample space where values of $p(x)$ are greater than or equal to λ . The λ -level set of the density function $p(x)$ denotes the set

$$S(p, \lambda) = \{x \in \mathbb{R}^m : p(x) \geq \lambda\}.$$

The set $S(p, \lambda)$ consists of q numbers of connected components $S_i(p, \lambda)$ that are pairwise disjoint, i. e.,

$$S(p, \lambda) = \bigcup_{i=1}^q S_i(p, \lambda)$$

with $S_i(p, \lambda) \cap S_j(p, \lambda) = \emptyset$ for $i \neq j$. The subsets $S_i(p, \lambda)$ are called λ -density clusters (λ -clusters for short). For various levels λ , we obtain a collection of λ -clusters that have a hierarchical structure: for any two λ -clusters A and B we have either $A \subset B$ or $B \subset A$ or $A \cap B = \emptyset$. The hierarchical structure is called *hierarchical density clusters*. The hierarchical density clusters is represented by a high density cluster tree. Each node of the high density cluster tree represents a λ -cluster. The

high density cluster tree is determined recursively. The root node represents the 0-cluster (entire data set). Children of a λ -cluster are determined by finding the lowest level μ such that $\mu > \lambda$ and λ -cluster has more than two μ -clusters. If we cannot find the value μ the λ -cluster is the leaf of the high density cluster tree. Otherwise, we can find q μ -clusters within the λ -cluster, and we create q children of the λ -cluster. Figure 3.3 shows the hierarchical density clusters of a bivariate density function. In Figure 3.3 (left), A is 0-cluster, B is 2-cluster, C and D are 3-clusters, E, F, and G are 5-clusters. In Figure 3.3 (right) we show the hierarchical density clusters of the bivariate density function that summarizes the structure of the bivariate density function.

In this section, we present two algorithms for computing hierarchical density clusters corresponding to the two methods for estimating the multivariate density function. The hierarchical density clusters are computed automatically without the need to choose any thresholds for the density level sets.

3.2.1 Hierarchical density clusters using histograms

Let the data set $X = \{X_i = (X_{i1}, \dots, X_{im}) : 1 \leq i \leq n\}$ have n data points in m dimensions. We transform the data into a unit hypercube $[0, 1]^d$ such that the scaled values \bar{X}_{ij} of X_{ij} all lie in the interval $[0, 1]$:

$$\bar{X}_{ij} = \frac{X_{ij} - \min_j}{\max_j - \min_j}, \quad j = 1, \dots, m; \quad i = 1, \dots, n,$$

where $\min_j = \min\{X_{ij} : 1 \leq i \leq n\}$, $\max_j = \max\{X_{ij} : 1 \leq i \leq n\}$, $j = 1, \dots, m$.

We partition the data space $[0, 1]^m$ into a set of cells. Each dimension can be divided into the same number of intervals. As the data set was transformed into the unit interval, we divide each dimension into N equally-sized portions. Hence, the size of the portions is

$$h = \frac{1}{N},$$

which is the edge length of the grid cells. Alternatively, one could omit the scaling step and operate with different bandwidths for the different dimensions.

As we showed in Section 3.1.2, the choices of the bandwidth parameter h ensure that

$$h \longrightarrow 0 \text{ and } nh^m \longrightarrow \infty,$$

which is consistent with the above criteria for a well-defined approximation of a density estimation. Unfortunately, we know little about the data set, i.e., the density function is unknown. The bandwidth that minimizes the AMISE in Equation (3.5) does not help to get the bandwidth h . A well-known method is Sturges' rule [Sco92] for selecting the size of cells in an univariate histogram case. The rule postulates

$$h = \frac{\text{range of samples}}{1 + \log_2(n)},$$

which for multidimensional data grids extend to

$$h = \frac{1}{\sqrt[m+4]{n}}.$$

We use this estimate as an indicator for our approach, but we also allow for manual adjustment.

The multivariate density function is estimated by Equation (3.2). As the area is equal for all cells, the density of each cell is proportional to the number of data points lying inside the cell. Without loss of generality, we can describe the density by the number of data points within each m -dimensional cell.

Each cell is given by

$$C(i_1, \dots, i_m) = C(i_1) \times \dots \times C(i_m), \quad i_j = 0, \dots, N-1, \quad j = 1, \dots, m,$$

where

$$C(i) = \begin{cases} [i h, (i+1) h) & \text{if } i = 0, \dots, N-2, \\ [i h, (i+1) h] & \text{if } i = N-1. \end{cases}$$

and indexed by a unique index $C(i_1, \dots, i_d)$.

Because of the “curse of dimensionality”, the data space is very sparse. The multivariate histogram density estimation is simple. However, in practice, there are several problems with this approach for high-dimensional data. In one dimension, there are N cells; in two dimensions, there are N^2 cells, and for data samples $x \in \mathbb{R}^m$ there are N^m cells. This exponential growth of the number of cells means that in high-dimensional spaces a very large amount of data is required to estimate the density. Typically, the sizes of data sets are significantly smaller than the numbers of grid cells generated. Therefore, there are many empty cells. In our approach, we do not store empty cells such that the amount of cells we are storing and dealing with is less than the size of the data set. Hence, operating on the grid instead of using the points does not increase the amount of storage but typically decreases it.

To create the nonempty cells, we use a partitioning algorithm that iterates through all dimensions. Assuming that we partition the k th dimension of the hyperrectangle $R = I_1 \times \dots \times I_m$ (where in the first $k-1$ dimensions the edges of R have length h and the edges in the remaining dimensions have length 1) into smaller hyperrectangles. Let $X_{1k}, \dots, X_{nk} \in \mathbb{R}$ denote values on the k th dimension of the n points inside the hyperrectangle R . These values fall into p ($p \leq N$) non-empty intervals $I_k^j, j = 1, \dots, p$, whose length are equal to h . Then, the hyperrectangle R is partitioned into N small hyperrectangles, of which p hyperrectangles $R_j = I_1 \times \dots \times I_k^j \times \dots \times I_m, j = 1, \dots, p$ are non-empty. We only store the non-empty ones. We apply this algorithm for each dimension to partition the data set into a set of non-empty cells of size h^m . Figure 3.4 shows partitioning in two dimensions. In Figure 3.4 (left), the first dimension is divided into $N = 4$ intervals. We obtain three non-empty rectangles, and one empty rectangle. Each non-empty rectangle is divided into $N = 4$ intervals as displaying in Figure 3.4 (right). We obtain 6 non-empty cells.

Optionally, this step of the algorithm can be adopted to deal with data sets containing noise by storing only those cells containing a number of data points larger than a noise threshold.

The time complexity for partitioning the data space into non-empty cells is $O(mn)$, i.e., the time complexity is linear in both the size of the data set and

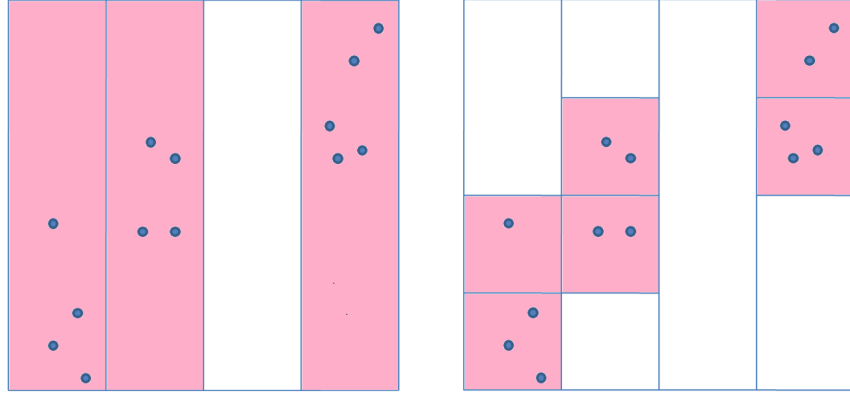


Figure 3.4: Memory-efficient partitioning of multidimensional space.

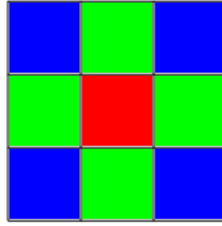


Figure 3.5: Neighborhood of the red cell for the two-dimensional space.

the number of dimensions. Hence, this algorithm is suitable for large-scale data sets of high dimensionality.

Given the partitioning into cells, we define that two cells $C(i_1, \dots, i_m)$ and $C(j_1, \dots, j_d)$ are neighbors, if $|i_k - j_k| \leq 1$ for all $k = 1, \dots, m$. Figure 3.5 shows all the neighbors of the red cell for the two-dimensional space. A cluster is represented by a connected region consisting of the union set of non-empty cells. We define a cell is a neighbor of a cluster if the cell is a neighbor of at least one cell belonging to the cluster. Two clusters are neighbors if there exists one cell belonging to one cluster that is a neighbor of another cell belonging to the other cluster.

We define a graph where the vertices represent the cells and the edges represent neighborhood information. Therefore, a cluster can be understood as a connected component of the graph. We apply the *breadth first search (BFS)* method to find connected components of the graph [CLRS01]. The procedure $BFS(V, E, v, C)$ considers the graph $G = (V, E)$ with set of vertices V and set of edges E , and finds the connected component C of G contains the vertex $v \in V$.

```

Procedure  $BFS(V, E, v, C)$ 
1  $C = \{v\}$ 
2 push  $v$  into a queue  $Q$ 
3 mark( $v$ )= visited
4 while  $Q$  is not empty
5      $v \leftarrow$  pop front  $Q$ 
6     for each vertex  $u \in V$  such that  $(u, v) \in E$ 
7         if mark( $u$ )=not-visited
8             push  $u$  into the queue  $Q$ 
9             mark( $u$ )=visited
10             $C \leftarrow C \cup \{u\}$ 
11        endif
12    endfor
13 endwhile

```

The procedure $ConnectedComponents(V, E, C = \{\})$ finds connected components of a graph $G = (V, E)$.

```

Procedure  $ConnectedComponents(V, E, C = \{\})$ 
1  $i = 1$ 
2 for each vertex  $v \in V$ 
3     if mark( $v$ )=not-visited
4          $BFS(V, E, v, C_i)$ 
5          $C \leftarrow C \cup \{C_i\}$ 
6          $i = i + 1$ 
7     endif
8 endfor

```

In order to find higher-density clusters within the detected clusters, we can apply a divisive hierarchical clustering method [HK06], where each cluster is split according to the density values. Given a detected cluster C , we remove all those cells from cluster C that contain the minimum number of data points. Then, we apply the graph-based cluster search algorithm from above to find again connected components in the reduced graph. The connected components represent the higher-density clusters within cluster C . If the number of higher-density clusters within C is $q \geq 2$, we create q new clusters (subclusters of C) and proceed with these (higher-density) clusters in the same manner. If $q = 1$, we iteratively remove further cells with a minimum number of data points. If we cannot find multiple higher-density clusters within a cluster, we call this cluster a *mode cluster*.

This procedure automatically generates a collection of density clusters \mathcal{T} that exhibit a hierarchical structure: for any two density clusters A and B in \mathcal{T} , we have $A \subset B$ or $B \subset A$ or $A \cap B = \emptyset$. This hierarchical structure is summarized by the density cluster tree (short: cluster tree). The root of the cluster tree represents all sample points.

Figure 3.6 (left) shows the histogram of a bivariate density estimation. The root node of a cluster tree contains all non-empty cells and finds the number of connected components of this node. We obtain two connected components. The

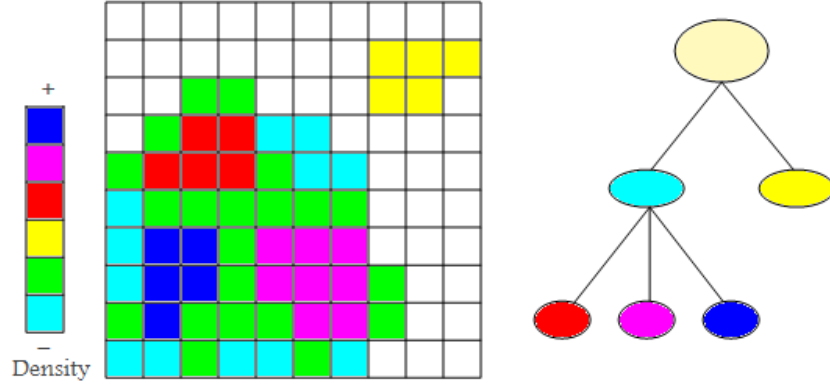


Figure 3.6: Cluster tree of density visualization with four modes shown as leaves of the tree. *Left*: Partitioning of the two-dimensional space. *Right*: A hierarchical density clusters.

two connected components are represented by two children of the root node (cyan and yellow). The yellow cluster is a homogeneous cluster. For the cyan cluster, we remove all cells that have minimum density (cyan cells). This cluster is still connected. We continue removing all cells that have minimum density (green cells). We obtain three connected components: the red, the blue, and the pink clusters. The three clusters are children of the cyan node. The three clusters are homogeneous clusters while the cyan cluster is a heterogeneous cluster. Figure 3.6 (right) shows a cluster tree with four mode clusters represented by the tree's leaves.

The cluster tree visualization provides a method to understand the distribution of data by displaying regions of modes of the multivariate density function. Each cluster contains at least one mode. The leaf nodes of the cluster tree are the mode clusters.

3.2.2 Hierarchical density clusters using kernels

Let the data set $X = \{X_i = (X_{i1}, \dots, X_{im}) : 1 \leq i \leq n\}$ have n data points in m dimensions. We recall the multivariate density function formula in Equation (3.7)

$$\hat{p}(x) = \frac{1}{nh_1 \dots h_m} \sum_{i=1}^n \prod_{j=1}^m K\left(\frac{x_j - X_{ij}}{h_j}\right),$$

for each $x \in \mathbb{R}^m$.

We consider a kernel function $K(x)$ with compact support $\text{supp}(K) = [-1, 1]$. For a point x , the density value $\hat{p}(x)$ in Equation (3.7) is obtained by summing up the contribution of all points X_i with $x_j - X_{ij} \in [-h_j, h_j]$ for all $j = 1, \dots, m$. We define the support of a point x as follows:

$$B(x) = B(x; h_1, \dots, h_m) = [x_1 - h_1, x_1 + h_1] \times \dots \times [x_m - h_m, x_m + h_m]. \quad (3.15)$$

The λ -level set of the density function $\hat{p}(x)$ can be estimated by a union of support of data points X_i such that $\hat{p}(X_i) \geq \lambda$:

$$S(\hat{p}, \lambda) = \bigcup_{X_i: \hat{p}(X_i) \geq \lambda} B(X_i; h_1, \dots, h_m).$$

This formula is derived from estimating the support of the conditional function $p_\lambda(x) = p(x|S(p, \lambda))$, i.e., the closed set $S(p_\lambda) = \overline{\{x \in \mathbb{R}^m : p_\lambda(x) > 0\}}$, where \overline{A} is the closure of the set A . The support of $p_\lambda(x)$ can be expressed as a union of small closed regions around the sampling observations of the conditional function $p_\lambda(x)$. The sampling observations of $p_\lambda(x)$ is not available, we use the observations X_i from the sampling observations of $p(x)$ such that $p(X_i) \geq \lambda$ as the sampling observations of $p_\lambda(x)$.

This λ -level set can be partitioned into some connected components and each connected component can be understood as a λ -cluster. A hierarchical density cluster is constructed based on the multivariate kernel density estimation and support of points. The three steps of our approach are given in the following:

- Multivariate kernel density estimation,
- Creating support of points,
- Clustering processes.

Kernel density estimation The value of the density function at point x is evaluated according to the formula in Equation (3.7). Because of the compact support of the kernel function, this formula can be written as:

$$\hat{p}(x) = \frac{1}{nh_1 \dots h_m} \sum_{X_i \in B(x)} \prod_{j=1}^m K\left(\frac{x_j - X_{ij}}{h_j}\right).$$

For example, the simplest kernel function is constant over its support (uniform kernel)

$$K(t) = \begin{cases} \frac{1}{2} & -1 \leq t \leq 1, \\ 0 & \text{otherwise.} \end{cases}$$

In this case, the density value of x can be estimated by the following

$$\hat{p}(x) = \frac{|B(x; h_1, \dots, h_m)|}{n \text{volume}(B(x; h_1, \dots, h_m))}, \quad (3.16)$$

where $|B(x; h_1, \dots, h_m)|$ is the number of observations in the region $B(x; h_1, \dots, h_m)$.

We use a kdtree data structure and a range search algorithm for finding all data points in a hyperrectangle $B(X_i; h_1, \dots, h_m)$ and computing the density value. A kdtree is a data structure for sorting a finite set of points from a multidimensional space [dBCvKO08]. We can construct a kdtree with the recursive procedure described below. In the procedure Build-kdtree(X, d), X is the set of data points, m is the dimension of the data set X , and d is the current splitting dimension.

Procedure Build-kdtree(X, d)

- 1 if X contains only one point
- 2 then return a leaf storing this point
- 3 else
- 4 split X at the median ℓ of the d th coordinate of the points in X .
- 5 let X_{left}, X_{right} be the set of points below and above ℓ .
- 6 $v_{left} \leftarrow \text{Build-kdtree}(X_{left}, (d+1) \bmod m)$
- 7 $v_{right} \leftarrow \text{Build-kdtree}(X_{right}, (d+1) \bmod m)$
- 8 create a node v storing ℓ with v_{left} and v_{right} are being the left and the right child of v .
- 9 return v .

To find data points that fall in a hyperrectangular region, we use the procedure Range-Search-kdtree(v, R), where v is the kdtree and R is the region.

Procedure Range-Search-kdtree(v, R)

- 1 if v is a leaf
- 2 then report the point stored at v if it lies in the range R
- 3 else if $\text{region}[\text{leftchild}(v)]$ is fully contained in R
- 4 then report $\text{subtree}(\text{leftchild}(v))$
- 5 else if $\text{region}[\text{leftchild}(v)]$ intersects R
- 6 then Range-Search-kdtree($\text{leftchild}(v), R$)
- 7 if $\text{region}[\text{rightchild}(v)]$ is fully contained in R
- 8 then report $\text{subtree}(\text{rightchild}(v))$
- 9 else if $\text{region}[\text{rightchild}(v)]$ intersects R
- 10 then Range-Search-kdtree($\text{rightchild}(v), R$)

Creating support of points The value $p(X_i)$ of the density function is computed for each data point x_i in the data set. The procedure to create the support of points is given in the following:

Procedure Creating-Supports($X = \{X_1, \dots, X_n\}, h_1, \dots, h_m$)

- 1 while X is not empty
- 2 find the maximum $X_j = \arg \max\{p(X_i) : X_i \in X\}$
- 3 create the support of data point X_j , $B(X_j; h_1, \dots, h_m)$
- 4 $X \leftarrow X \setminus (X \cap B(X_j; h_1, \dots, h_m))$
- 5 endwhile

This procedure generates a set of support points from highest to lowest density. The number of supports that are generated is smaller than the number of data points.

Clustering processes In the last step, we only consider the support points that are generated in the creating support points step. The order (according to the density value of the center point of the support) is automatically given. Without loss of generality, let $B(X_1), \dots, B(X_k)$ be the supports of the data points used in

the preceding step with $p(X_1) \geq \dots \geq p(X_k)$ and $k \leq n$. Now, let's assume that the clusters C_1, \dots, C_q have already been generated when considering the support $B(X_i)$.

- If $B(X_i)$ is not intersecting with any cluster $C_j \in \{C_1, \dots, C_k\}$, we create a new cluster C_{q+1} :

$$C_{q+1} = \{B(X_i)\}.$$

- If there is exactly one cluster $C_j \in \{C_1, \dots, C_k\}$ with $B(X_i) \cap C_j \neq \emptyset$, then we merge $B(X_i)$ with C_j :

$$C_j \leftarrow C_j \cup B(X_i).$$

- If there are ℓ clusters C_1, \dots, C_ℓ ($\ell \leq q$) with $B(X_i) \cap C_j \neq \emptyset, j = 1, \dots, \ell$, the region $B(X_i)$ is a valley in the density function that connects the ℓ clusters C_1, \dots, C_ℓ . We create a new cluster that contains the ℓ clusters and $B(X_i)$:

$$C_{q+1} = \left(\bigcup_{j=1}^{\ell} C_j \right) \cup B(X_i).$$

Cluster C_{q+1} contains ℓ clusters C_1, \dots, C_ℓ (hierarchical structure), which do not need to be further considered during the algorithm. The updated list of clusters contains $C_{\ell+1}, \dots, C_q, C_{q+1}$.

Procedure Clustering($B(X_1), \dots, B(X_k), C = \{\}$)

```

1 initialize  $q = 1$ ,  $C_1 = \{B(X_1)\}$ , and add  $C_1$  to  $C$ 
2 for  $i = 2$  to  $k$  do
3     intersects= $\emptyset$ 
3     for  $j = 1$  to  $q$  do
4         if  $B(X_i) \cap C_j \neq \emptyset$ 
5             intersects  $\leftarrow$  intersects  $\cup \{j\}$ 
6     endfor
7     if intersects= $\emptyset$ 
8         create a new cluster  $C_{q+1} = \{B(X_i)\}$ ,  $q = q + 1$ , and
          add  $C_{q+1}$  to  $C$ 
9     else if intersects= $\{j_1\}$ 
10         $C_{j_1} \leftarrow C_{j_1} \cup \{B(X_i)\}$ 
11    else intersects= $\{j_1, \dots, j_\ell\}$ , ( $\ell \geq 2$ )
12        create a new cluster  $C_{new} = \left( \bigcup_{j=j_1}^{j_\ell} C_j \right) \cup \{B(X_i)\}$ 
13        remove  $C_{j_1}, \dots, C_{j_\ell}$  from  $C$ ,  $q = q - \ell + 1$ , and
          add  $C_{new}$  to  $C$ .
14 endfor
```

Figure 3.7 illustrates the clustering processes using a 2D example. First, a set of support of points are generated in Figure 3.7 (left), the blue support contains 5

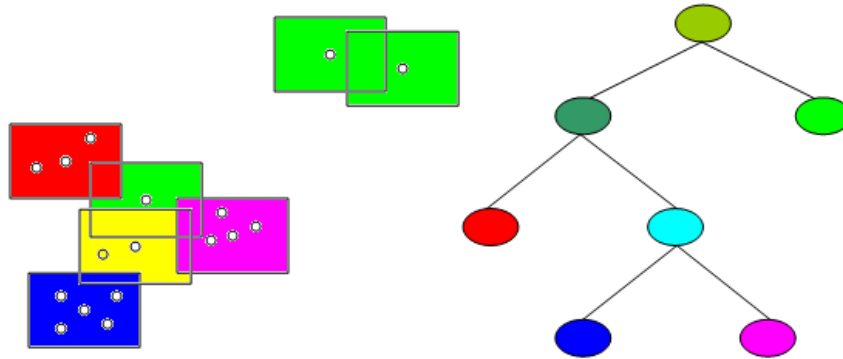


Figure 3.7: Cluster tree of density visualization with 4 modes shown as leaves of the tree.

points, the magenta support contains 4 points, the red support contains 3 points, the yellow support contains 2 points, and the green support contains 1 point. Second, we take the support with the highest density (blue), and we create a cluster that contains this support. Next, we take the second largest density support (magenta). It does not intersect with previous clusters, such that we create a new cluster that contains the magenta support. Similarly, we create a new cluster that contains the red support. Next, we take the largest density support among the remaining supports (the yellow support) this support connects the blue support with the magenta support. Hence, the two clusters that contain these supports are merged into a new cluster (cyan clusters), as we can see in Figure 3.7 (right). We continue the procedure with the remaining supports and finally obtain a hierarchical density cluster.

3.2.3 Discussions and comparisons

We have been presenting two hierarchical density cluster algorithms based on non-parametric density estimations. Both algorithms can be applied to large-scale and high dimensional data sets. One main advantage of the two hierarchical density clusters is to automatically identify density clusters without any threshold density level set parameters. Another advantage of the two hierarchical density clusters is that it is suitable for data sets containing noise. Both algorithms are very intuitive and easy to implement.

In the hierarchical density clusters using histograms, the multidimensional data space is partitioned into non-overlapping cells, while the hierarchical density clusters uses kernels to divide the multidimensional data space into overlapping supports of data points. The hierarchical density clusters using histograms apply a top-down strategy, while the hierarchical density clusters using kernels apply a bottom-up strategy. In our experiments, the number of support of data points generated by kernel method is smaller than the number of non-empty cells in histogram method. In general, experiments on the size of cells and support of points are desirable. The hierarchical density clusters using kernels can be applied to gene expression data, while the hierarchical density clusters using histograms cannot generate density

clusters for such data.

Our partitioning of the multidimensional data space into non-overlapping regions in the histogram method seems to be similar to the hybrid clustering method [Won82]. The author uses the k -means algorithm to partition the multidimensional data set. The data set is reduced to the k mean data points of the k clusters. A single-linkage clustering algorithm is applied using matrix distances (the distance between two nearest mean points is determined by the inverse of the density of the middle point of the two points). Clusters are identified by the maximal sets of the k clusters whenever the distance is smaller than a threshold parameter. However, it is hard to interpret the density level sets of the clusters. Moreover, the clusters are not automatically determined and the density estimation is not consistent.

Wong and Lane [WL83] improved the hybrid clustering by using the k nearest neighbors for density estimations. Similar to the hybrid clustering, a single-linkage clustering algorithm is applied using matrix distances (the distance between two neighbors is defined by summing the inverse of the density values of the two points). However, the density level sets of clusters are not obvious in this hierarchical structure and the algorithm is not practicable for large-scale data sets.

Stuetzle [Stu03] constructs high density clusters based on runt size of the minimal spanning tree. Clusters are automatically identified and density level sets of clusters are very intuitive. The disadvantage of this algorithm is that the density estimation is not consistent, as the nearest neighbors are used for density estimation. Also, the computational costs for the minimal spanning tree for large-scale high-dimensional data are high.

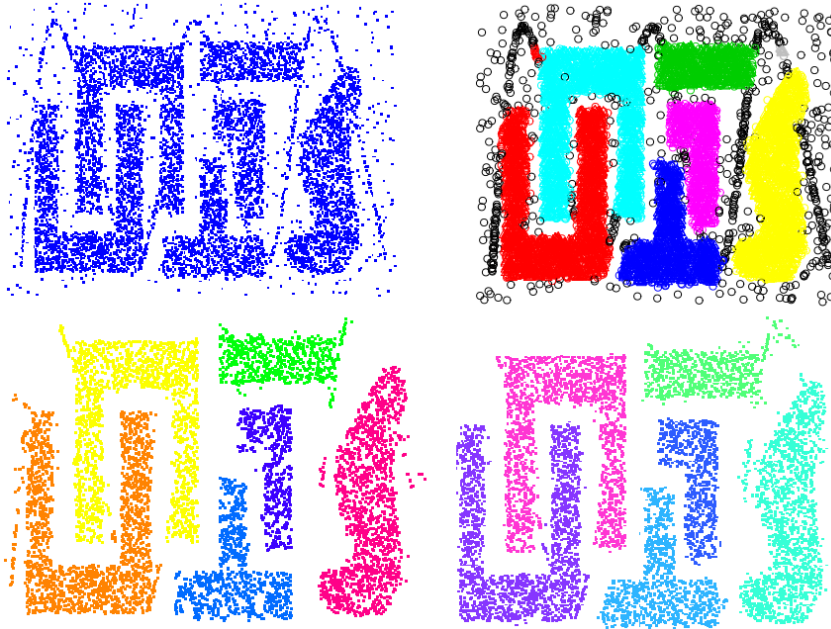


Figure 3.8: Clustering result for data set *t4.8k.dat*. *Upper left*: Original data set. *Upper right*: DBSCAN clustering result. *Lower left*: Histogram-based clustering result. *Lower right*: Kernel-based clustering result.

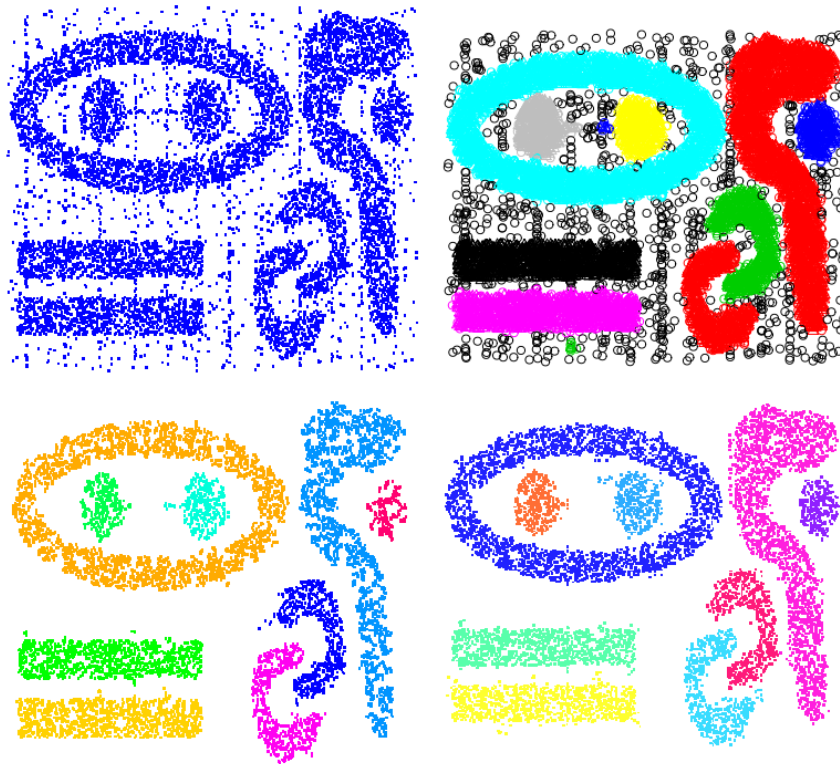


Figure 3.9: Clustering result for data set *t7.10k.dat*. *Upper left*: Original data set. *Upper right*: DBSCAN clustering result. *Lower left*: Histogram-based clustering result. *Lower right*: Kernel-based clustering result.

In this section, we evaluate the performance of our histogram-based method and kernel-based method and compare the results with the DBSCAN approach [EKSX96] (using the implementation embedded in the system R). Synthetic data sets are used in the experiments: *t4.8k.dat* and *t7.10k.dat* [KHK99]. As DBSCAN cannot extract cluster hierarchies, we chose data where no such hierarchies are present. The clustering results for *t4.8k.dat* and *t7.10k.dat* are shown in Figure 3.8 and Figure 3.9. Different colors are used to indicate the different clusters and clusters are represented by connected regions. Clusters with different sizes, shapes, and densities are identified.

Both data sets were transformed into the unit square $[0, 1] \times [0, 1]$ in a pre-processing step. DBSCAN produces the clustering results that are shown in Figure 3.8 (upper right) and Figure 3.9 (upper right) with $Eps = \frac{1}{50}$ and $MinPts = 10$. The histogram-based method produces the clustering results shown in Figure 3.8 (lower left) and Figure 3.9 (lower left) by dividing each dimension into $N = 50$ intervals and removing all cells containing less than 10 points. The kernel-based method produces the clustering results shown in Figure 3.8 (lower right) and Figure 3.9 (lower right) with the kernel size $h = \frac{1}{50}$ and removing all supports of points that contain less than 10 points. Figure 3.8 and Figure 3.9 show that both the histogram-based method and the kernel-based method can detect arbitrary shapes

of clusters as DBSCAN algorithm. The clustering results are sensitive with respect to the parameters Eps in the DBSCAN algorithm, number of bins N in the histogram-based method, and kernel size h in the kernel-based method.



Figure 3.10: The sensitivity of clustering results with respect to the grid size in the histogram-based method (*left*) and the kernel size in the kernel-size method (*right*).

The histogram-based method depends on the grid size and the kernel-based method depends on the kernel size. Figure 3.10 (left) shows the relationship of the number of bins and the number of homogeneous clusters for the “Out5d” data set. Figure 3.10 (right) shows the relationship of the kernel sizes and the number of homogeneous clusters for the “Out5d” data set. In practice, a good choice for the parameters can be determined empirically by simple running the algorithms multiple times with different parameters. Those parameters are selected when the number of homogeneous clusters start exhibiting slight differences only. The red lines in Figure 3.10 is good values for the grid size and the kernel size.

Data sizes	10k	20k	30k	40k	50k	60k
dbscan	23	91	213	404	672	1019
kernel	54	115	166	211	291	367
histogram	1	1	2	2	2	2

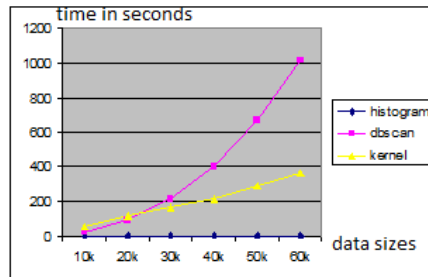


Figure 3.11: Computation times (in seconds) of DBSCAN, histogram-based method, and kernel-based method for different data sizes.

Our experiments were performed on a Linux system with 4GB RAM memory and a 2.66 GHz processor. The computation times are shown in Figure 3.11. The computation times of the histogram-based method are much better than those of the DBSCAN and the kernel-based method. For small data sizes (up to 20,000 points), DBSCAN and the kernel-based method are in the same range, but for large data sizes, the kernel-based method is more efficient than the DBSCAN method.

Chapter 4

Nested level set visualization of hierarchical density clusters

In this chapter, we present a method for projecting multidimensional data into visual space (two or three dimensions) with optimized star coordinates. The optimized star coordinate system minimizes the overlap of separated mode clusters in the hierarchical density clusters described in Chapter 3. The star coordinates encoding displays the multidimensional data as point clouds that contain the hierarchical structure of the clusters. The large amount of displayed data points make it difficult to detect the hierarchical structure. We propose to visualize the hierarchical density clusters such that each cluster is enclosed by a contour or a surface with minimum volume. Therefore, the hierarchical density clusters are visualized by nested contours or nested surfaces.

4.1 Optimized star coordinates

A standard method of visualizing multidimensional data is to reduce its dimensionality to two or three dimensions. Dimensionality reduction can be seen as the process of transforming data from a high-dimensional space to a low-dimensional subspace in such a way that the transformation ensures the maximum possible preservation of information or minimum reconstruction error. The dimensionality reduction problem can be formulated as follows:

Dimensionality reduction Let $X = \{x_1, \dots, x_m\}$ be a set of n data points in an m -dimensional space, i.e., $x_i \in \mathbb{R}^m$, then a dimensionality reduction technique tries to find a corresponding output set of patterns $Y = \{y_1, \dots, y_n\}$ in a d -dimensional space, i.e., $y_i \in \mathbb{R}^d$, where $d < m$ and Y provides the most faithful representation of X in the lower dimensional space.

Dimensionality reduction techniques can be classified into linear methods and non-linear methods.

Linear methods These methods reduce dimensionality by performing linear transformations on the input data and trying to find a globally defined flat subspace.

These methods are most effective if the input patterns are distributed more or less throughout the subspace. Some linear dimension reduction methods [GKWZ07] are independent component analysis (ICA), principal component analysis (PCA), factor analysis (FA), and projection pursuit (PP).

Non-linear methods When the input patterns lie on or near a low-dimensional sub-manifold of the input space the structure of the data set may be highly non-linear, and linear methods are likely to fail. Non-linear methods, on the other hand, try to find the locally defined flat subspace by non-linear transformations. Non-linear dimension reduction methods [GKWZ07] include kernel principal component analysis (KPCA), principal curves and surfaces, locally linear embedding (LLE), Hessian LLE, Laplacian eigenmaps (LE), latent tangent space alignment (LTSA), ISOMAP, multidimensional scaling (MDS), self-organizing map (SOM), and generative topographic mapping (GTM).

The linear dimension reduction method can be reformulated as follows: we want to find a best d -dimensional subspace for representing high-dimensional data set in some sense. We assume that the basis of the d -dimensional subspace is given by

$$\begin{cases} v_1 &= (v_{11}, v_{12}, \dots, v_{1m}) \\ &\vdots \\ v_d &= (v_{d1}, v_{d2}, \dots, v_{dm}) \end{cases} \quad (4.1)$$

and a linear transformation corresponding to this basis can be understood as a mapping

$$\begin{aligned} P : \mathbb{R}^m &\rightarrow \mathbb{R}^d \\ x &\mapsto Px = (\langle x, v_1 \rangle, \dots, \langle x, v_d \rangle) \end{aligned} \quad (4.2)$$

where $\langle x, y \rangle = \sum_{j=1}^m x_j y_j$ denotes the scalar product. This mapping can be rewritten as:

$$\begin{aligned} P(x) &= (\langle x, v_1 \rangle, \dots, \langle x, v_d \rangle) \\ &= \left(\sum_{j=1}^m x_j v_{1j}, \dots, \sum_{j=1}^m x_j v_{dj} \right) \\ &= \sum_{j=1}^m x_j (v_{1j}, \dots, v_{dj}), \end{aligned}$$

or

$$P(x) = \sum_{j=1}^m x_j a_j, \quad (4.3)$$

where

$$\begin{cases} a_1 &= (v_{11}, v_{21}, \dots, v_{d1}), \\ &\vdots \\ a_m &= (v_{m1}, v_{m2}, \dots, v_{md}). \end{cases} \quad (4.4)$$

The m vectors $\{a_1, \dots, a_m\}$ in d -dimensional space are called *star coordinates*.

The linear mapping in Equation (4.3) is contractive, i.e., the Euclidean distance between two points in visual space is always smaller than the Euclidean distance between the two points in multidimensional space. Therefore, two points in the multidimensional space are projected to visual space preserving the similarity properties of clusters. In other words, the mapping does not break clusters. Still, clusters may overlap. In star coordinates, the m vectors $\{a_1, \dots, a_m\}$ in d -dimensional space are represented for m dimensions of multidimensional data sets. That allows us to interpret the results, i.e., if the value x_j of a point in multidimensional data is high then the mapping of the point will close to the vector a_j .

The standard $2D$ star coordinates system [Kan00] for visualizing m -dimensional points is given by:

$$a_i = \left(\cos \frac{2\pi(i-1)}{m}, \sin \frac{2\pi(i-1)}{m} \right), \quad i = 1, \dots, m.$$

and the standard $3D$ star coordinates [AdO04] is an extension of $2D$ star coordinates to

$$a_i = \left(\cos \frac{2\pi(i-1)}{m}, \sin \frac{2\pi(i-1)}{m}, 1 \right), \quad i = 1, \dots, m.$$

In order to visualize the high density clusters in a way that allows clusters to be correlated with all m dimensions, we need to use a coordinate system that incorporates all m dimensions. Such a coordinate system can be obtained by using star coordinates. In star coordinates the result of a linear transformation is plotted, which maps multidimensional data to the d -dimensional subspace.

Assume that a hierarchy of high-density clusters has q mode clusters, which do not contain any higher-level densities. Let m_i be the mean or barycenter of the points within the i th cluster, $i = 1, \dots, q$. Our objective is to choose a linear transformation that maintains distances between the q mode clusters. In the described linear transformation, two points in multidimensional space are projected to star coordinates preserving the similarity properties of clusters. In other words, the mapping of multidimensional data to a low-dimensional space determined by Equation (4.2) does not break clusters. The additional goal of optimized star coordinates is to separate mode clusters as much as possible.

Let $\{v_1, \dots, v_d\}$ be an orthonormal basis of the candidate d -dimensional subspace of a linear transformation. The mean m_i is mapped to a point z_i ,

$$z_i = P(m_i) = (\langle m_i, v_1 \rangle, \dots, \langle m_i, v_d \rangle), \quad i = 1, \dots, q.$$

The desired choice of a star coordinate layout is to maximize the distance of the q points z_1, \dots, z_q . This can be obtained by picking d vectors $\{v_1, \dots, v_d\}$ of the d -dimensional subspace such that the objective function

$$\sum_{1 \leq i < j \leq q} \|z_i - z_j\|^2 = \sum_{1 \leq i < j \leq q} \|Vm_i - Vm_j\|^2 = \text{trace}(V^t S V)$$

is maximized, where $V = [v_1, \dots, v_d]^t$ and S is the covariance matrix

$$S = \sum_{1 \leq i < j \leq q} (m_i - m_j)(m_i - m_j)^t = q \sum_{i=1}^q (m_i - \bar{m})(m_i - \bar{m})^t$$

with $\bar{m} = \frac{1}{q} \sum_{i=1}^q m_i$ being the mean of the q barycenters of the mode clusters.

Hence, the d vectors v_1, \dots, v_d are the d unit eigenvectors corresponding to the d largest eigenvalues of the matrix S . The optimized star coordinates $\{a_1, \dots, a_m\}$ are obtained from $\{v_1, \dots, v_d\}$ as in Equation (4.4) and the multidimensional data points are mapped into visual space as in Equation (4.3).

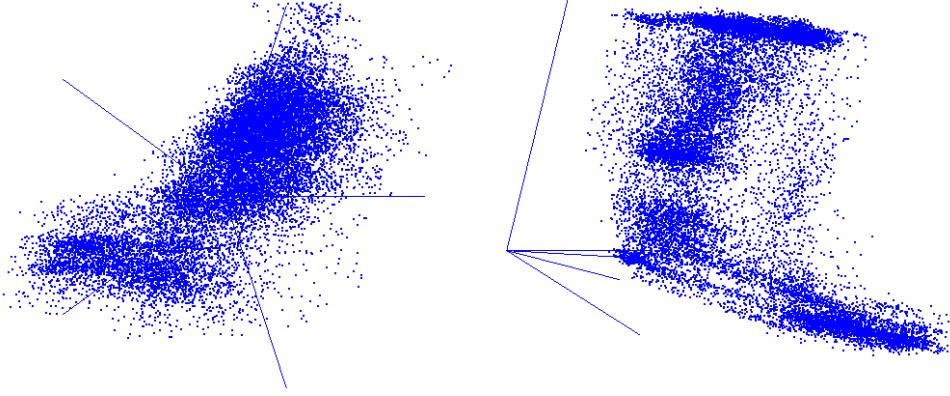


Figure 4.1: Visualization of the “out5d” data set with 2D star coordinates. (Left) Standard 2D star coordinates. (Right) Optimized 2D star coordinates.

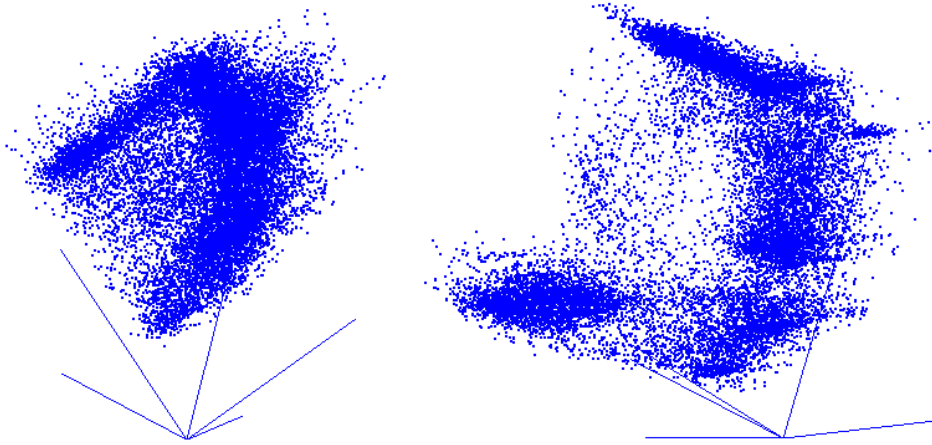


Figure 4.2: Visualization of the “out5d” data set with 3D star coordinates. (Left) Standard 3D star coordinates. (Right) Optimized 3D star coordinates.

We show star coordinate representations for visualizing the well-known data set “out5d” that contains 16,384 points with five dimensions. Figure 4.1 presents 2D star coordinates visualizing the multidimensional data. Figure 4.1 (left) shows the standard 2D star coordinates and Figure 4.1 (right) shows the optimized 2D star coordinates. Similarly, Figure 4.2 (left) shows the standard 3D star coordinates and Figure 4.2 (right) shows the optimized 3D star coordinates. In both figures, the optimized star coordinates reveal more intuitive representations of clusters than standard star coordinates. However, we still have difficulties to identify hierarchical

density clusters. Clusters are extracted by contours in $2D$ and by surfaces in $3D$ conveying the hierarchical density cluster structure of the multidimensional data sets.

4.2 Cluster enclosure

When large multidimensional data sets are projected to star coordinates and plotted as point clouds, it is difficult to detect clusters in the point cloud. To support the understanding of hierarchical density clusters, we propose to visualize clusters as connected regions in visual space. A region of a cluster contains all points in this cluster, it has a minimal area or volume, and regions of clusters maintain the hierarchical structure of clusters.

Consider a cluster described by a set of points

$$P = \{p_i = (p_{i1}, \dots, p_{id}) : i = 1, \dots, n\}$$

in $2D$ or $3D$ visual space e.g. $d = 2$ or $d = 3$. To find a connected and compact region that contains all points in P , we consider points in P as independent observations of an unknown density function, and describe the region as the support of this density function. A support of the density function can be estimated based on its observations p_1, \dots, p_n as

$$S(r) = \bigcup_{i=1}^n B(p_i, r), \quad (4.5)$$

where r is a positive parameter and $B(p_i, r) = \{p : \|p - p_i\| \leq r\}$ is the closed ball centered at p_i with radius r . The parameter r is estimated such that $S(r)$ is still connected, i.e.,

$$\bar{r} = \inf \left\{ r > 0 : S(r) = \bigcup_{i=1}^n B(p_i, r) \text{ is a connected set} \right\}.$$

The value \bar{r} is a half of the length of the longest edge of the Euclidean minimum spanning tree (EMST) of P . However, the boundary of the set $S(r)$ is not smooth.

We use density field functions to approximate $S(r) = \bigcup_{i=1}^n B(p_i, r)$ by a region with smooth boundary.

The basic idea is that each spherical region $B(p_i, r)$ is generated by a density field function. The density field function has its maximum at the point and decreases around the point. The density field function only influences the region within the radius $R = 2r$. An implicit function is given by the sum of all density field functions and $S(r)$ is approximated by choosing an iso-value of the implicit function and taking the region bounded by the respective contour. Figure 4.3 shows the process of approximating $S(r)$ for two points. Figure 4.3 (left) shows the two density field functions, Figure 4.3 (middle) shows the implicit function and a line representing the iso-value, and Figure 4.3 (right) shows a contour that approximates the boundary of $S(r)$.

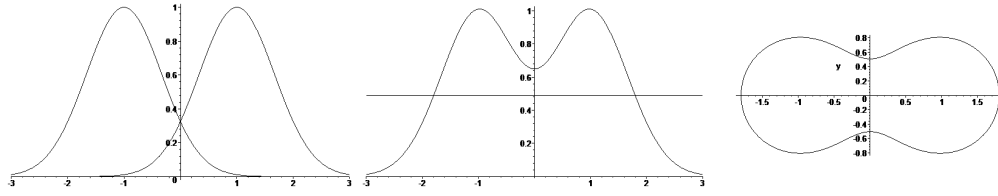


Figure 4.3: The process of approximating $S(r)$ for two points. *Left*: The two density field functions. *Middle*: The implicit function. *Right*: The region approximation of $S(r)$.

4.2.1 Euclidean minimum spanning tree

A spanning tree of a graph G is a subgraph T that is connected and acyclic. The Euclidean minimum spanning tree (EMST) is a minimum spanning tree of a set of points, where the weight of the edge between each pair of points is the distance between those two points. An EMST connects a set of points using edges such that the total length of all the edges is minimized and any points can be reached from any other by following the edges. The EMST of a set P with n points is a connected graph having $(n - 1)$ edges.

Euclidean minimum spanning tree A spanning tree $T = (P, E)$ of the set of points P is an EMST of P if $\sum_{(p_i, p_j) \in E} \|p_i - p_j\|$ is minimized over all spanning trees of P .

Kruskal's algorithm is an algorithm in graph theory that finds a minimum spanning tree for a connected weighted graph. We apply this algorithm to a complete graph of the set of points P to find the EMST. First, edges of the complete graph are sorted by weight (distance). Second, we iteratively take and remove one edge from a queue that stores all edges in the given order. Next, we use a disjoint-set data structure to keep track of which vertices are in which components. Using this data structure, we can decide whether the current edge is an edge of the EMST. The time complexity of Kruskal's algorithm is $O(n^2 \log n)$ to find the EMST of n points. This algorithm is given in the following:

```

Procedure Kruskal ( $G = (P, E)$ )
1  for each point  $p_i$  in  $P$  do
2    define an elementary cluster  $C(p_i) = i$ 
3  sort edges in a queue  $Q$ 
4  define tree  $T \leftarrow \emptyset$ 
5  while  $T$  has fewer than  $(n - 1)$  edges do
6     $e = (p_i, p_j) \leftarrow Q.\text{front}()$ 
7    if  $C(p_i) \neq C(p_j)$  then
8      add the edge  $e = (p_i, p_j)$  to  $T$ 
9       $C(p_i) = C(p_j) \leftarrow \min(C(p_i), C(p_j))$ 
10 return  $T$ .

```

4.2.2 Density field functions

The most commonly used visualization of the implicit function is the sum of a spherically decreasing density field around points. Most of the effort in constructing these primitives goes into creating a flexible distance function. The properties of the distance function will determine the shape of primitives and how they blend together. The implicit function can be written in the following form:

$$f(p) = \sum_{i=1}^n D(r_i),$$

where $r_i = \|p - p_i\|$ is the distance from point p to a location at the point p_i , and $D(r)$ is a density field function.

Bloppy models Blinn [Bli82] introduced a point-based implicit model. The density field is a Gaussian function centered at each point. The parameters a and b be used the standard deviation and the height of the function respectively. The influence of the point is

$$D_1(r) = b \exp(-ar^2).$$

Metaball models A variation of Blinn's model is the metaball model. The metaball model uses a piecewise quadratic instead of an exponential function.

$$D_2(r) = \begin{cases} 1 - \frac{3r^2}{R^2} & \text{if } 0 \leq r \leq \frac{R}{3} \\ \frac{3}{2} \left(1 - \frac{r}{R}\right)^2 & \text{if } \frac{R}{3} \leq r \leq R \\ 0 & \text{otherwise} \end{cases}$$

where R is the maximum distance that the control primitives contribute to the density field function.

Soft objects Wyvill et al. [WMW86] simplified the calculation somewhat by defining a cubic polynomial based on the radius of influence for a particle and the distance from the center of the particle to the field location. The key is that the influence must be 1 when the distance r is 0, and 0 when the distance is equal to the radius R of influence. A function which satisfies these requirements is

$$D_3(r) = \begin{cases} 2\left(\frac{r}{R}\right)^3 - 3\left(\frac{r}{R}\right)^2 + 1 & \text{if } r \leq R \\ 0 & \text{otherwise.} \end{cases}$$

The computation of this equation is somewhat slow, due to the square root calculation in the distances. The authors replaced the density field function $D_3(r)$ by

$$D_4(r) = \begin{cases} -\frac{4}{9}\left(\frac{r^2}{R^2}\right)^3 + \frac{17}{9}\left(\frac{r^2}{R^2}\right)^2 - \frac{22}{9}\left(\frac{r^2}{R^2}\right) + 1 & \text{if } r \leq R \\ 0 & \text{otherwise.} \end{cases}$$

For simplicity, we use the density function given by

$$D_5(r) = \begin{cases} (1 - \frac{r^2}{R^2})^2 & \text{if } r \leq R \\ 0 & \text{otherwise} \end{cases}$$

or

$$D_6(r) = \begin{cases} (1 - \frac{r^2}{R^2})^3 & \text{if } r \leq R \\ 0 & \text{otherwise.} \end{cases}$$

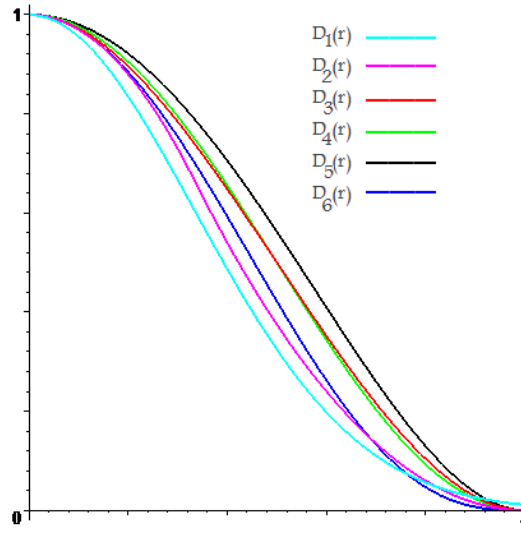


Figure 4.4: The density field functions.

Figure 4.4 shows different kinds of density field primitives. All the density functions are smooth. The support of the Gaussian function $D_1(r)$ is not compact, while the support of the other density field functions is compact. The compact support of the density field function is advantageous for the computation of the implicit function.

An implicit contour can be described by a set of generator points P , where each generator point $p_i \in P$ has a radius of influence R . The influence of a single generator point p_i at a point p is described by a density function $D(r_i) = D(\|p - p_i\|)$. The summation of all the density functions for all generator points form the density field $f(p)$. Thus, a level set of the implicit function $f(p) = c$ is defined as those points p where the sum of the density values of all generators are equal to the threshold value $c \geq 0$. We choose the two parameters describing the radius of influence R and the iso-value c such that we can guarantee that

$$S(R, c) = \{p : f(p) \geq c\}$$

is connected, contains all points in P , and has an area or volume of minimum extension.

Let p_i and p_j denote two points, which are connected by the longest edge in the EMST of the set of points P as in Subsection 4.2.1. We choose the radius of influence by $R = \|p_i - p_j\|$ and the iso-value can be selected such that $S(R, c)$ contains the line segment

$$[p_i, p_j] = \{p = tp_i + (1 - t)p_j : 0 \leq t \leq 1\},$$

which ensure that $S(R, c)$ is connected. This leads to

$$f\left(\frac{p_i + p_j}{2}\right) \geq D\left(\left\|\frac{p_i + p_j}{2} - p_i\right\|\right) + D\left(\left\|\frac{p_i + p_j}{2} - p_j\right\|\right) = 2D\left(\frac{R}{2}\right),$$

and the iso-value c can be selected as $2D\left(\frac{R}{2}\right)$.

4.2.3 Enclosing point clouds

In this subsection, we describe a method to enclose point clouds by a connected domain in two or three dimensions. A linear blend is defined by summing distance functions to each point element. The blending equation for a set of primitive point elements p_i is

$$f(p) = \sum_i D(\|p - p_i\|).$$

We compute a bounding box of all points in a cluster that contains the set of points $P = \{p_i = (x_{i1}, \dots, x_{id}) : i = 1, \dots, n\}$. We denote this bounding box by

$$B = [a_1, b_1] \times \dots \times [a_d, b_d].$$

We divide the bounding box B into a number of cells with the length of each dimension being equal to h , and set $n_i = \lceil \frac{b_i - a_i}{h} \rceil$, $i = 1, \dots, d$. We compute the value at each grid (i_1, \dots, i_d) , $i_j = 1, \dots, n_j$, $j = 1, \dots, d$, of the function $f(p)$. Initially, the value of $f(i_1, \dots, i_d)$ is zero.

Procedure Computing($f(i_1, \dots, i_d)$)

- 1 for each point element p_i do
- 2 find support region of $D(\|p - p_i\|)$
- 3 for each grid point that is indexed (i_1, \dots, i_d) belonging to the support of $D(\|p - p_i\|)$
- 4 $f(i_1, \dots, i_d) + D(\|(a_1 + i_1h, \dots, a_d + i_dh) - p_i\|)$

Marching squares Each vertex is marked inside if the value of the vertex is greater or equal than the iso-value and marked outside if the value of the vertex is smaller than the iso-value. A square has four vertices, each vertex has two cases (inside or outside), and we have $2^4 = 16$ cases of squares as shown in Figure 4.6. Each square is indexed by

$$\text{index}(v) = \begin{array}{|c|c|c|c|} \hline v_4 & v_3 & v_2 & v_1 \\ \hline \end{array}$$

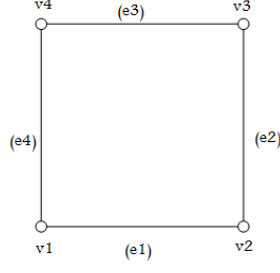


Figure 4.5: The index of a square.

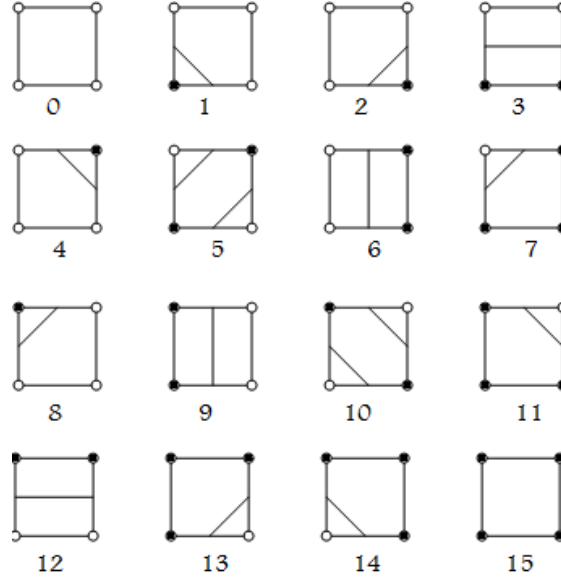


Figure 4.6: The 16 case intersection of contour with square.

where $v_i = 1$ if vertex v_i is marked inside and $v_i = 0$ if vertex v_i is marked outside. An edge intersection of a square is indexed by $\text{index}(e)$.

$$\text{index}(e) = \begin{bmatrix} e_4 & e_3 & e_2 & e_1 \end{bmatrix}$$

where $e_i = 1$ if the edge e_i is intersecting with the boundary and $e_i = 0$ if the edge e_i lies inside or outside the domain. Figure 4.5 shows the indexed vertices and edges of a square. An intersection point of the edge with the boundary is computed using linear interpolation: Considering edge (v_1, v_2) with $f_1 = f(v_1) < f_{iso}$ and $f_2 = f(v_2) > f_{iso}$, the intersection point is given by $v = (1 - t)v_2 + tv_1$, where $t = \frac{f_2 - f_{iso}}{f_2 - f_1}$.

Marching cubes Similar to marching squares, marching cubes finds intersections of the surface with a cube. There are $2^8 = 256$ ways the surface may intersect the cube, and the symmetries reduce those 256 cases to 15 patterns, see Figure 4.7. We use a lookup table for 256 cases for surface intersection with the edges of the cube.

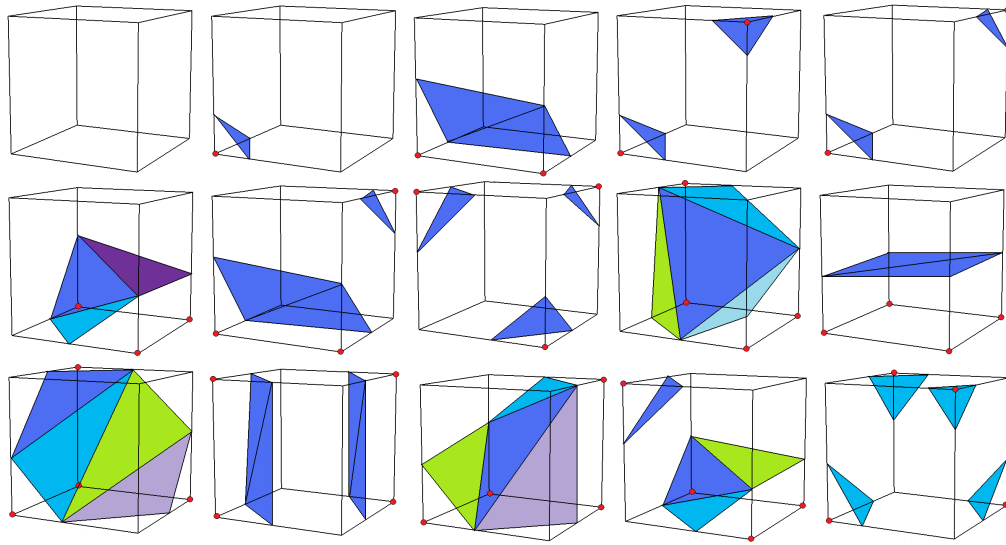


Figure 4.7: The 15 basic intersection topologies.

4.3 Results and discussions

We applied our methods to five data sets. The first data set is one of the most well-known data sets in pattern recognition literature, the so-called Iris data set. We have chosen this data set, as it can be compared to existing results from pattern recognition. The Iris data set has 150 points in only four dimensions. The four dimensions represent the four features sepal length (SL), sepal width (SW), petal length (PL), and petal width (PW). Previous analyses have found out that there are three categories in the data set, namely the iris setosa, the iris versicolor, and the iris virginica, each having 50 patterns or points. We divide each dimension of the data space into $N = 10$ steps leading to a grid size of 10^4 . We obtain 110 non-empty cells and compute the high density areas as described in Section 3.2.1.

Figure 4.8 shows our results when visualizing nested density clusters with 2D star coordinates. Figure 4.9 (left) shows our results when visualizing nested density clusters with 3D star coordinates. For the lowest density, the data visualization shows two clusters. The first cluster includes the versicolor and virginical group, while the setosa group has been separated. The setosa group is homogeneous, i.e., it does not exhibit subclusters. For higher density, the versicolor and virginical group is split into three groups (two versicolor groups and one virginical group) such that one can observe the groups represented in the data set. Figure 4.9 (right) shows the setosa group in parallel coordinates, which demonstrates the homogeneity of the extracted cluster. The visualization of this group exhibits high values in dimension SW and low values in all other attributes. This correlation to the dimensions also becomes obvious in the star coordinates layout. Hence, we observe that the groups known from prior analyses were also separated by our approach and this separation is intuitive using our visualization method. Moreover, we noticed that the versicolor and virginical groups are more closely related than the setosa group. The individual

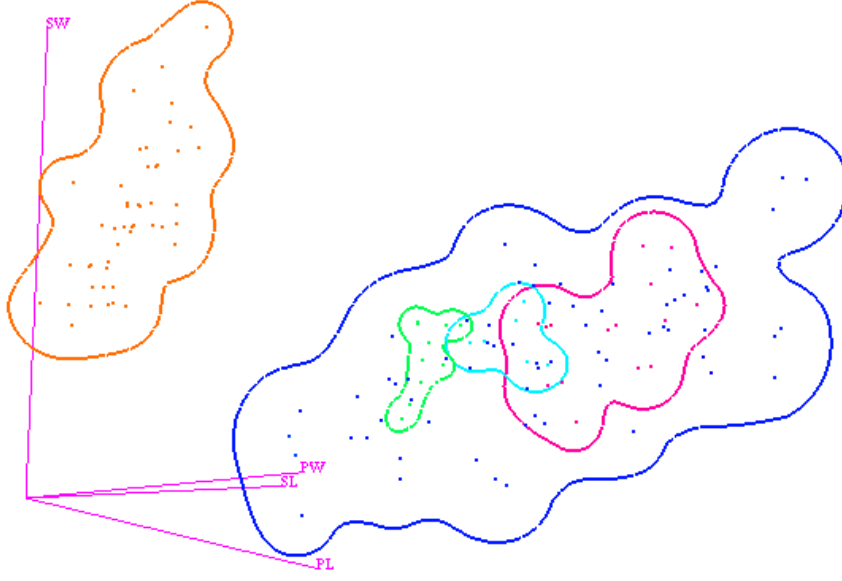


Figure 4.8: Visualization of four-dimensional Iris data. Nested density cluster visualization based on the hierarchical density cluster tree using our 2D star coordinates.

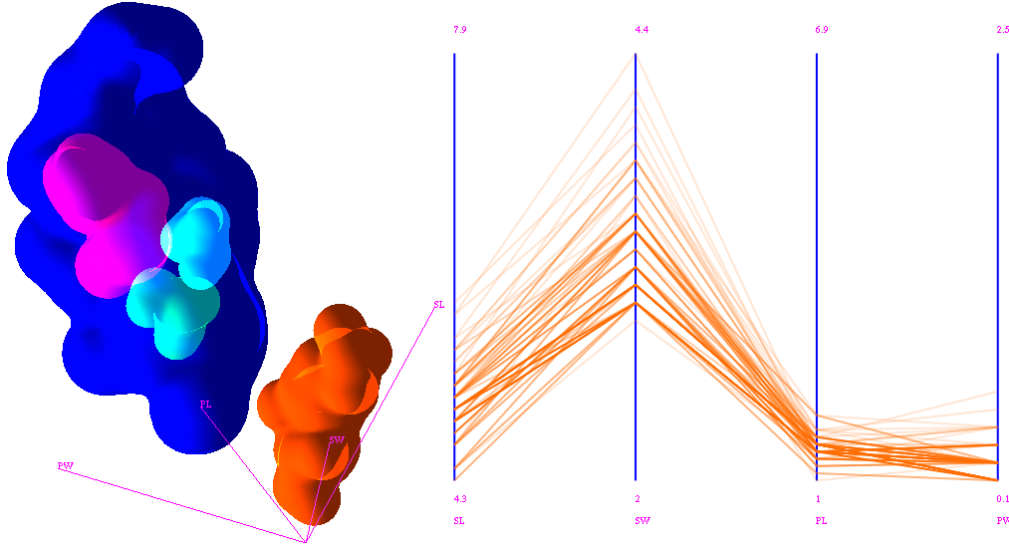


Figure 4.9: Visualization of four-dimensional Iris data. *Left*: Nested density cluster visualization based on the cluster tree using our 3D star coordinates. *Right*: Setosa group cluster (right cluster) is selected and its homogeneity is evaluated using parallel coordinates. In both pictures, the relation between the setosa group cluster with the dimension can be observed.

modes can be distinguished easily in 2D, there is a significant overlap of the cyan and the magenta cluster, and the shapes of the surfaces represent the shapes of the projected clusters, enclosing all projected cluster points in a narrow fashion. We

can observe that the versicolor and virginical groups are the least distant ones.

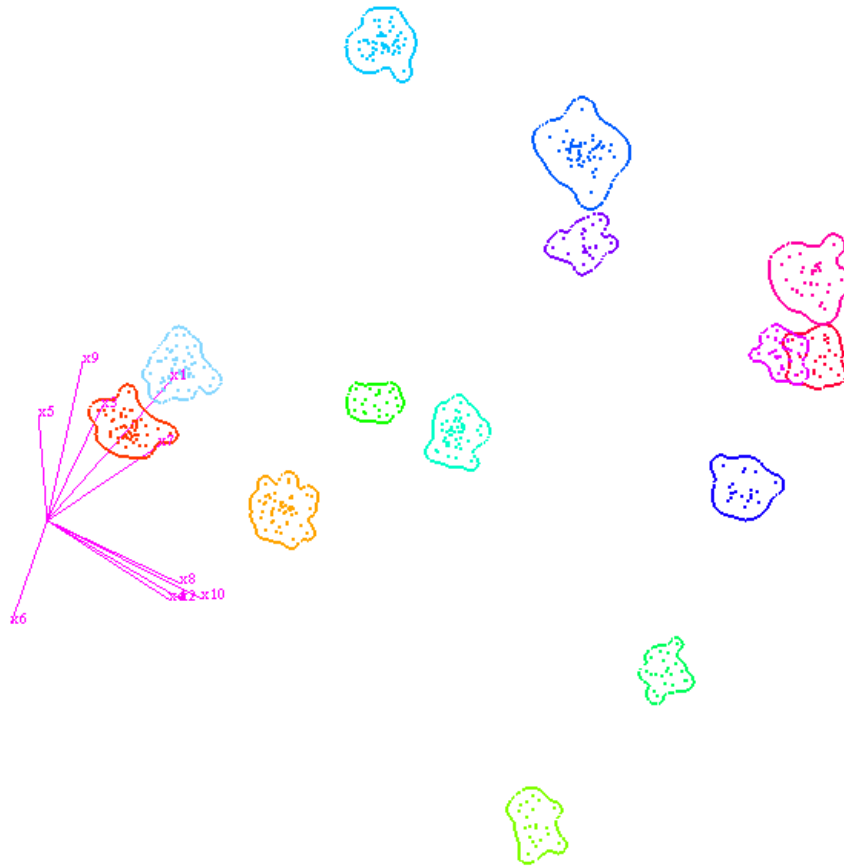


Figure 4.10: Visualization of ten-dimensional synthetic data with optimized 2D star coordinates.

The second data set, we have been investigating to evaluate our methods is a synthetic data set with 480 records in a 10-dimensional space. Again, we divide each dimension of the data space into $N = 10$ steps leading to a grid size of 10^{10} . We obtain 468 non-empty cells and compute the high density areas as described in Section 3.2.1. Figure 4.10 shows the high density cluster visualization in 2D star coordinates and Figure 4.11 (left) shows the high density cluster visualization in 3D star coordinates. The distribution of the clusters is rather spread, while the density at the modes is similar throughout the data set. In Figure 4.11 (right), we show the high density visualization using standard 3D star coordinates. It can be observed that several clusters overlap when using standard 3D star coordinates, while our optimized layout manages to avoid overlaps. The star coordinates display 14 clusters obviously, while Figure 1.5 only shows five clusters in scatterplot matrix and Figure 1.6 only shows one cluster in parallel coordinates.

Figure 4.12 shows the 3D star coordinates visualizing another synthetic data set. This data set has 38,850 observations in a 20-dimensional space, of which 14,831 observations define 8 clusters and the remaining 24,019 observations produce noise with a uniform distribution. We divide each dimension of the data space into $N = 10$

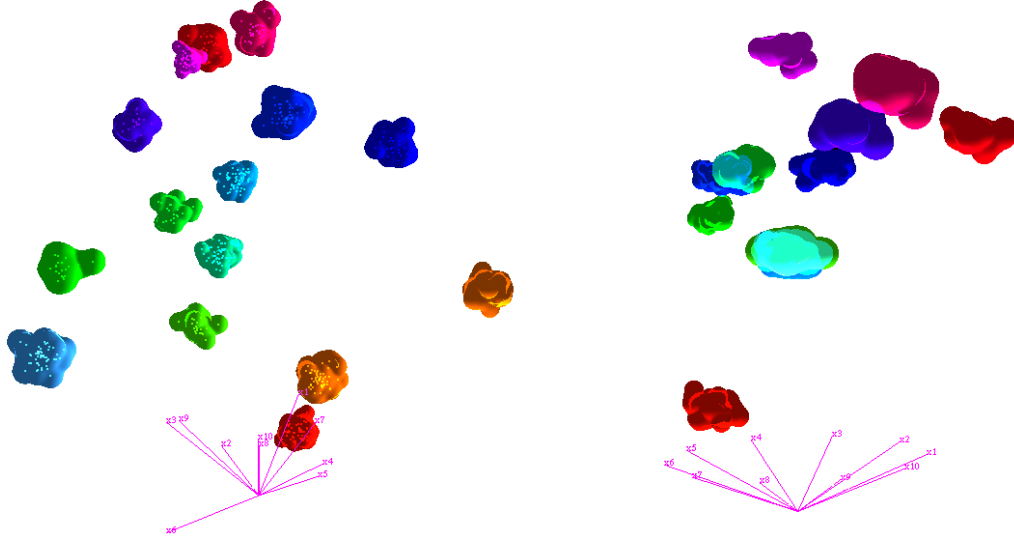


Figure 4.11: Visualization of ten-dimensional synthetic data. *Left*: Nested density cluster visualization based on the cluster tree using our optimized 3D star coordinates. *Right*: Nested density cluster visualization based on the cluster tree using standard 3D star coordinates. The standard approach does not avoid overlapping clusters.

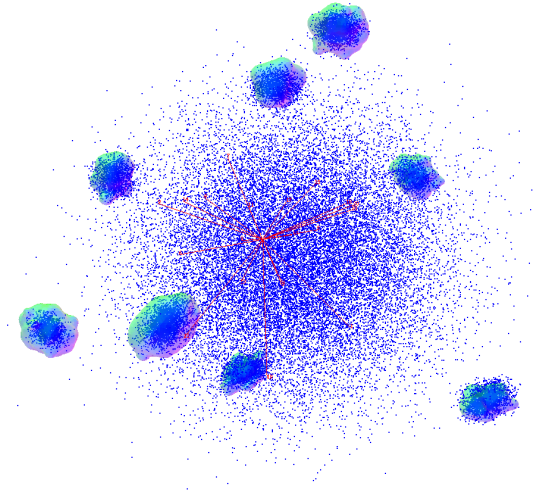


Figure 4.12: Visualization of 20-dimensional synthetic data using optimized 3D star coordinates.

steps leading to a grid size of 10^{20} . We obtain 1075 non-empty cells after removal of all noisy cells (i.e. the ones containing only one point) and compute the high density areas as described in Section 3.2.1. The data shows eight clusters of sizes 2107, 1038, 2085, 1312, 1927, 1329, 1581 and 1036, respectively.

In Figure 4.13 (left), these clusters are rendered by contours of groups of points

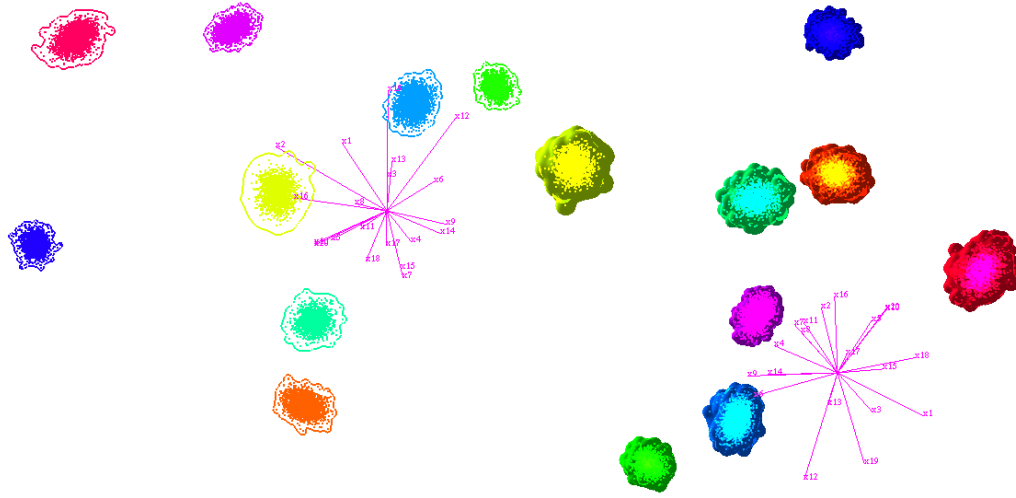


Figure 4.13: Visualization of twenty-dimensional synthetic data using star coordinates. *Left*: The high density cluster visualization using our optimized 2D star coordinates. *Right*: The high density cluster visualization using our optimized 3D star coordinates.

in a two-dimensional space. The groups are non-overlapping and homogeneous. In Figure 4.13 (right), we show these clusters by wrapping surfaces using our 3D star coordinates. In both 2D and 3D star coordinates, we can identify eight clusters.

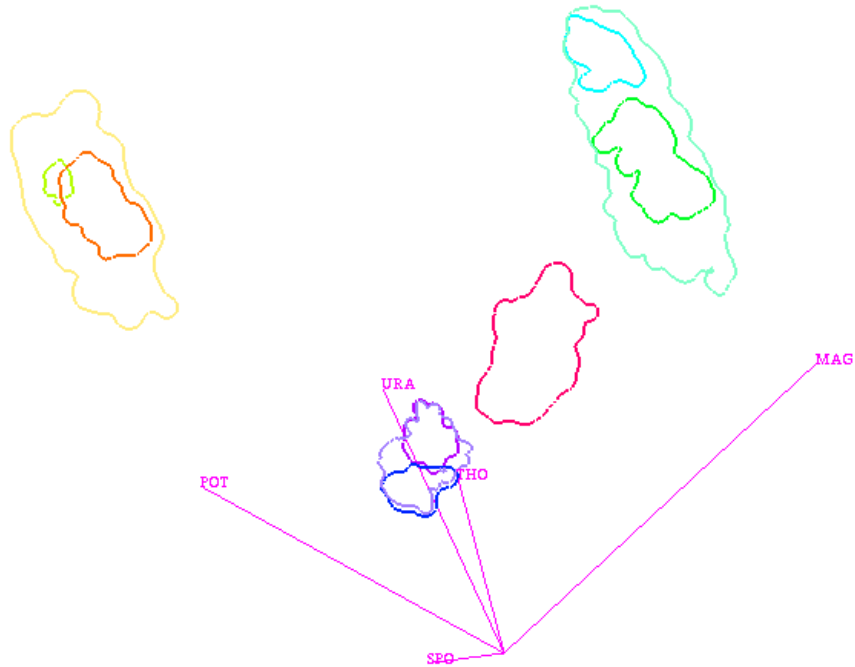


Figure 4.14: Visualization of five-dimensional out5d data using an optimized 2D star coordinates.

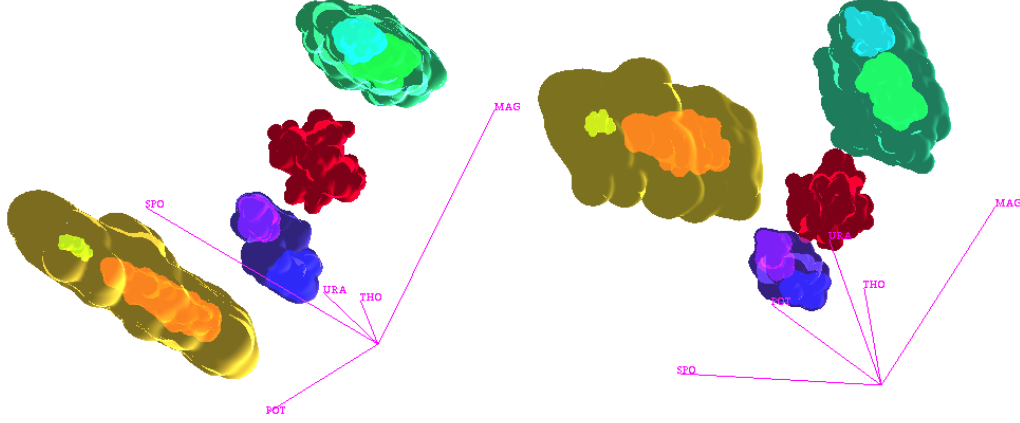


Figure 4.15: Visualization of five-dimensional out5d data using an optimized 3D star coordinates.

The fourth data set, we consider is a real data set called “out5d”. It contains 16,384 data points with 5 attributes: spot (SPO), magnetics (MAG), potassium (POS), thorium (THO), and uranium (URA). We divide each dimension of the data set into $N = 10$ equally-sized intervals and only keep the non-empty cells that contain more than 50 points. For the lowest level of density, we obtain four clusters. One of the clusters is homogeneous, while each of the other clusters contain two higher density clusters. In Figure 4.14, we show the hierarchical density clusters by nested contours with the optimized 2D star coordinates. Using only two visual dimensions, not all clusters could be separated. The subclusters of the yellow cluster overlap. In Figure 4.15, we show the respective 3D result. Using three visual dimensions there are no overlapping clusters. Figure 1.7 shows this data set in parallel coordinates, we cannot identify any clusters. Figure 4.1 displays this data set with 2D star coordinates and Figure 4.2 displays this data set with 3D star coordinates. In both figures, we can identify some clusters of the data set. However, the hierarchical density clusters are lost.

The last data set, we consider is an astronomy data set of a White Dwarf obtained by a multi-field smoothed particle hydrodynamics simulation. The simulation makes use of seven elements. Each particle stores the mass fractions of the seven chemical elements Helium, Carbon, Oxygen, Neon, Magnesium, Silicon, and Iron. The size of the data set is about 500,000 points. We divide each dimension of data space into $N = 10$ steps without prior scaling (as mass fraction values should be preserved). In Figure 4.16 (left), we visualize the clusters of the astronomical data set at time $t = 45$ using 3D star coordinates. We obtain a cluster tree with three mode clusters. The optimized 3D star coordinates indicate immediately that only Helium (red), Silicon (black), and Carbon (green) play a significant role in the simulation. The use of the star coordinates allow us to correlate the data to the dimensions. The first mode cluster is highly dependent on Helium, the second mode cluster is highly dependent on Silicon, and the smaller third mode cluster lies between them. The last two clusters are more closely related, which is indicated by the lower-density cluster

that includes them. In Figure 4.16 (right), we show the astronomical data set at another time step $t = 5$. At this earlier time step, the cluster tree has the same topology but Oxygen (blue) still plays a more important role. These observations were all feasible to the domain scientists.

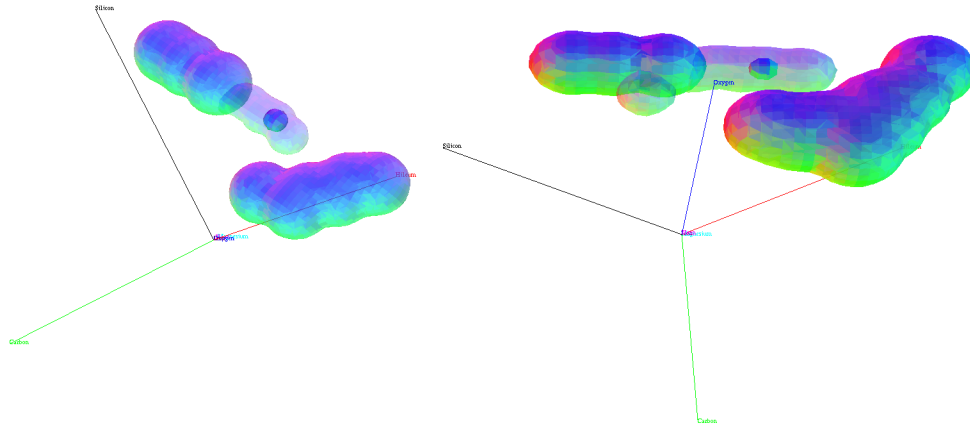


Figure 4.16: Visualization of seven-dimensional astrophysical data using the optimized 3D star coordinates. *Left*: time step 45. *Right*: time step 5.

We have presented a method to visualize the hierarchical density clusters using optimized star coordinates. The hierarchical density clusters are constructed in the multidimensional data space. The multidimensional data sets are projected into visual space by linear mapping. The advantage of linear mapping keeps the hierarchical structure of clusters. We obtained the hierarchical density clusters in visual space. Clusters are connected regions in multidimensional data. In visual space, the clusters are display as connected regions. The connected regions are extracted by using the minimal spanning tree of all points within the clusters. The regions represent the multidimensional clusters containing all points within the clusters while having minimal areas or volumes. An implicit function is used to extract the regions' boundary as contours or surfaces. The implicit function is a summation of all spherical density field functions of points within the cluster. The iso-value of the implicit function is selected such that the region where the implicit function is greater than the iso-value contains the longest edge of the minimal spanning tree. This assures that cluster regions are connected. The advantage of using the minimal spanning tree for this purpose is that it allows to represent any shape of the cluster. The generated contours or surfaces adapt to the spatial distribution of the points within the clusters. However, the computation of the minimal spanning tree is high.

The Narcissus system [HDWB95] encloses clusters in visual space by spheres. Hence, the points' distribution within the clusters is not very well represented. H-blob [SBG00] combines clustering and visualizing a hierarchical structure. In the visualizing step, a cluster is extracted by an implicit function. The implicit function of this cluster is considered a summation of ellipsoidal density field functions, i.e., subclusters of this cluster are represented by ellipsoids. The iso-value of the implicit

function is computed by the minimal field value of the interconnecting lines between the outliers and the cluster center, i.e., it is likely to be star-shaped. In the clustering step, single linkage hierarchical clustering is used. Therefore, the implicit function is a summation of two ellipsoidal density field functions. The assumption of ellipsoidal shape for the subclusters is a limitation of the h-blob. However, the computation of the iso-value and the surface extraction is fast.

Star coordinates [SY06, SY07, CL04, TM03, STTX08] are used for projecting the multidimensional data into a visual space and clusters are identified in the visual space. This is different from our approach, clusters are identified in the multi-dimensional space. The optimized star coordinates is similar with class-preserving projection [DMS98, DMS02], but the star coordinates was not mentioned in [DMS98, DMS02].

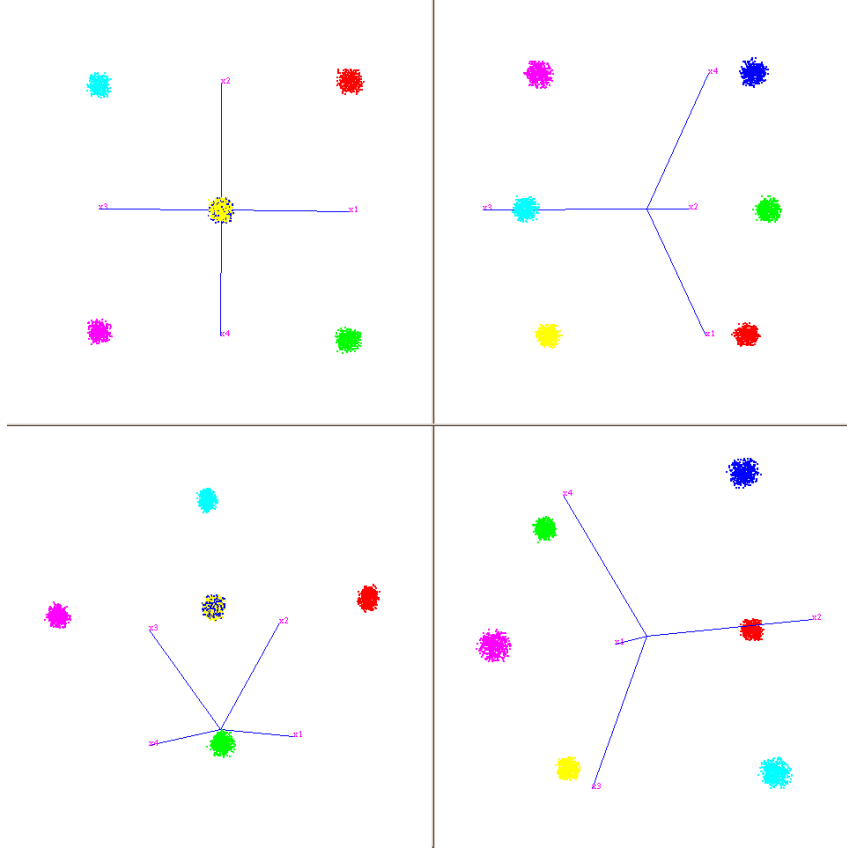


Figure 4.17: The 4D data set. *Upper left:* 2D standard star coordinates. *Upper right:* 2D optimized star coordinates. *Lower left:* 3D standard star coordinates. *Lower right:* 3D optimized star coordinates.

In this section, we evaluate our optimized star coordinates and compare the results with those obtained by standard star coordinates with optimal order of the coordinates axes. We consider the synthetic 4D data set with four attributes. The 4D data set contains six clusters with cluster centroids $(1, 1, 0, 0)$, $(1, 0, 1, 0)$, $(1, 0, 0, 1)$, $(0, 1, 1, 0)$, $(0, 1, 0, 1)$, and $(0, 0, 1, 1)$. Figure 4.17 (upper left) shows the data set in

2D standard star coordinates and Figure 4.17 (lower left) shows the data set in 3D standard star coordinates. With any reordering of standard star coordinates, two of the six clusters are still overlapping. With 2D optimized star coordinates (Figure 4.17 upper right) and 3D optimized star coordinates (Figure 4.17 lower right), we can obviously see all the six clusters.

Next, we consider a method for measuring the quality of views on multidimensional data sets. Given a data set $X = \{x_i \in \mathbb{R}^m : 1 \leq i \leq n\}$ and a cluster structure $C(X) = \{1, \dots, q\}$ defining q clusters. Let c_i be the centroid of the i th cluster ($1 \leq i \leq q$), and let $x \in X$ with a cluster label $label(x) = i$ that indicates that x belongs to the i th cluster. Sips et al. [SNLH09] introduced the Centroid Distance (CD) to measure the compactness and separation of clusters in a multidimensional data space. A low-dimensional embedding capturing this basic property should also show separated clusters. CD describes the property of cluster members that the distance $dist(x, c_i)$ to its cluster centroid should always be minimal in comparison to the distance to all other cluster centroids, thus:

$$dist(x, c_i) < dist(x, c_j) \quad 1 \leq j \leq q, j \neq i,$$

where $dist$ is the Euclidean distance. $CD(x, c_i) = true$ denotes that the centroid property for x and its centroid c_i is fulfilled. Let $X' = \{x'_i \in \mathbb{R}^d : 1 \leq i \leq n\}$ be a projection of X into a visual space ($d = 2$ or $d = 3$). Distance Consistency (DSC) is defined as

$$DSC(X') = \frac{\#\{x' : CD(x', c'_{label(x)}) = true\}}{n} \cdot 100,$$

where x' is the projection of the data point x and c'_i is the centroid of the i th cluster in the visual space. Table 4.1 shows the DSC for two synthetic data sets, called $5D$ and $6D$, comparing all reordering of 2D (3D) standard star coordinates to find the best DSC and 2D (3D) optimized star coordinates. The $5D$ data set has five clusters with five attributes and the $6D$ data set has 14 clusters with six attributes. Figure 4.18 and Figure 4.19 show the best reordering with 2D standard star coordinates (upper left), 2D optimized star coordinates (upper right), the best of reordering with 3D standard star coordinates (lower left), and 3D optimized star coordinates (lower right). Table 4.1 shows that the optimized star coordinates are superior to finding the optimal reordering with standard star coordinates, as the DSC values obtained with optimized star coordinates are always higher (or equal, if the standard star coordinates achieve a perfect separation).

Data	Best 2D reordering	2D optimized	Best 3D reordering	3D optimized
5D	95.62	97.86	97.88	98.75
6D	82.71	100	100	100

Table 4.1: Distance consistency measure to compare the quality of the projection with the best reordering of the standard star coordinates and optimized star coordinates. Optimized star coordinates obtain better values.

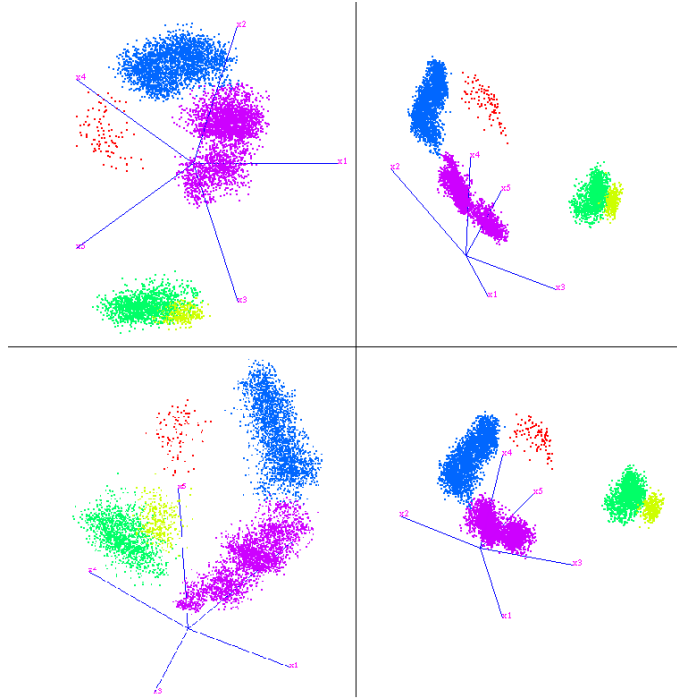


Figure 4.18: The 5D data set. *Upper left:* The best reordering of 2D standard star coordinates. *Upper right:* 2D optimized star coordinates. *Lower left:* The best reordering of 3D standard star coordinates. *Lower right:* 3D optimized star coordinates.

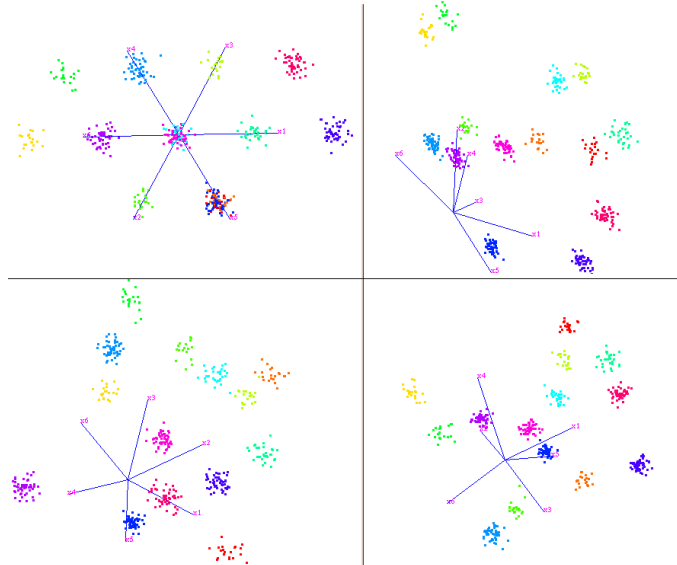


Figure 4.19: The 6D data set. *Upper left:* The best reordering of 2D standard star coordinates. *Upper right:* 2D optimized star coordinates. *Lower left:* The best reordering of 3D standard star coordinates. *Lower right:* 3D optimized star coordinates.

Chapter 5

Interactive visual exploration of hierarchical density clusters

In this chapter, we present an interactive tool for the visual exploration of hierarchical density clusters. To visually represent the cluster hierarchy, we present a $2D$ radial layout that supports an intuitive understanding of the distribution and structure of the multidimensional multivariate data set. Individual clusters can be explored interactively using parallel coordinates when being selected in the cluster tree. Furthermore, we integrate circular parallel coordinates into the radial hierarchical cluster tree layout, which allows for the analysis of the overall cluster distribution. We apply an automatic coloring scheme based on the $2D$ radial layout of the hierarchical cluster tree using hue, saturation, and value of the HSV color space.

5.1 Radial layout of density cluster hierarchy

Based on the hierarchical density clusters using histograms or kernels in Chapter 3, we present a layout for visualizing the resulting cluster tree. Of course, the visualization techniques that are described in this section and the subsequent ones can be applied to any hierarchical clustering result of multidimensional multivariate data.

Our visualization is based on drawing the hierarchical tree structure in a radial layout. A radial drawing is a variation of a layered drawing where the root of the tree is placed at the origin and layers are represented as concentric circles centered at the origin [TBET99].

Let n be the number of leaves and $m + 1$ be the depth of the hierarchical tree structure. The fundamental idea for our tree drawing is as follows: Considering a unit circle, the leaf nodes are placed evenly distributed on that unit circle, the root node is placed at the origin of the circle, and the internal nodes are placed on circular layers (with respect to the same origin) whose radii are proportional to the depth of the internal nodes. Hence, all mode clusters are represented by nodes placed on the unit circle. These clusters are homogeneous. All other clusters are represented by nodes placed on layers within the unit circle. These clusters are heterogeneous.

For the placement of internal nodes of the cluster tree, we use the notation

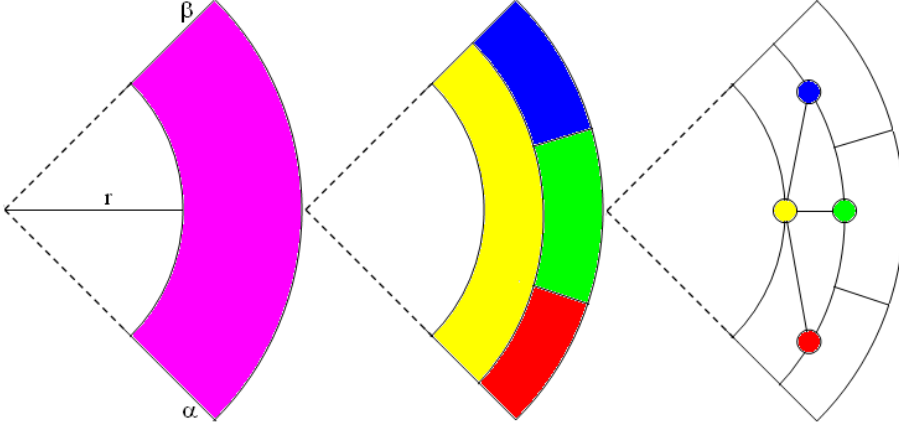


Figure 5.1: Radial layout of a cluster tree. (Left) An annulus wedge domain $W = (r, \alpha, \beta)$. (Middle) Splitting the annulus wedge for placing three subtrees. (Right) Placing internal nodes of cluster tree.

of an *annulus wedge*. Given a polar coordinate representation, an annulus wedge $W = (r, \alpha, \beta)$ denotes an unbounded region that lies outside a circle with a center at the origin and radius r and is restricted by the two lines corresponding to angles α and β . Figure 5.1 (left) shows an annulus wedge $W = (r, \alpha, \beta)$ (restricted to the unit circle).

Let tree T be the subtree of our cluster tree that is to be placed in the annulus wedge $W = (r, \alpha, \beta)$. The radius r denotes the distance of the root of T to the origin. If the root of T has depth d in the entire cluster tree, then $r = \frac{d}{m}$. Moreover, we use the notation $\ell(T)$ for the number of leaves of a tree T . Now, let T_1, \dots, T_k be those subtrees of tree T , whose root is a child node of T . For each subtree T_i , we compute the annulus wedge $W_i = (r_i, \alpha_i, \beta_i)$, where $r_i = \frac{d+1}{m}$ is the radius for placing the root node T_i ,

$$\alpha_i = \alpha + \sum_{j < i} \ell(T_j) \frac{2\pi}{n},$$

and

$$\beta_i = \alpha_i + \ell(T_i) \frac{2\pi}{n}.$$

Figure 5.1 (middle) shows how an annulus wedge is split for a tree T with three subtrees T_1, T_2, T_3 . This iterative splitting of the annulus wedge is started at the root node of the cluster tree, which is represented by an annulus wedge $(0, 0, 2\pi)$.

Finally, we can position all internal nodes of the cluster tree within the respective annulus wedge. Considering subtree T with the corresponding annulus wedge $W = (r, \alpha, \beta)$, we place the node at position $\left(r \cos \frac{\alpha + \beta}{2}, r \sin \frac{\alpha + \beta}{2}\right)$ with respect to the polar coordinate system. Figure 5.1 (right) shows the placement of nodes for the annulus wedges shown in Figure 5.1 (middle).

The 2D radial layout above can suffer from edges crossing problems. To avoid this problem, we reposition the interior nodes of the tree structure. Assume that

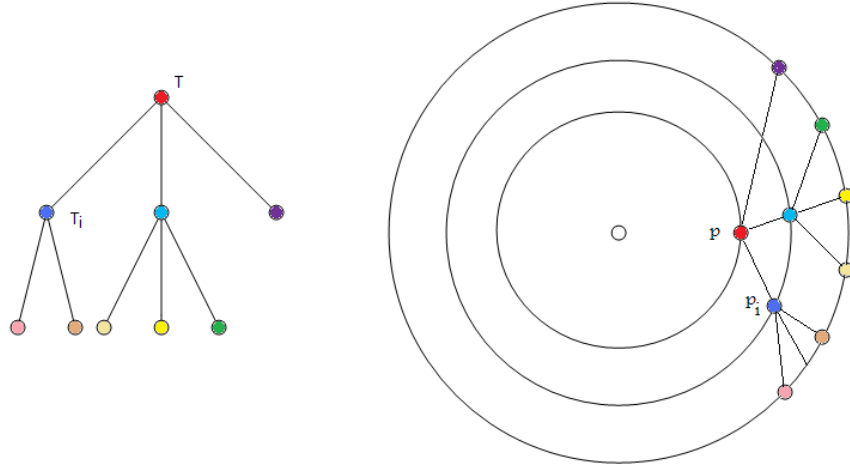


Figure 5.2: Radial layout of hierarchical structure without edge crossing.

the root of a subtree T has the position p in a unit circle and lies on the circle with radius $r = \frac{d}{m}$. Moreover, let T_i be a subtree whose root is a child of the root of T . The position p_i of the root of subtree T_i is the intersection of the circle with radius $r = \frac{d+1}{m}$ and a line segment from p to the middle point of the positions of the leaf nodes of subtree T_i on the unit circle. Figure 5.2 (left) shows the subtree T and its child T_i . The root node of T is displayed on the radial layout by the position p and then its child T_i by p_i in Figure 5.2 (right). Starting with the root node at the origin of the unit circle, we find the position of each interior node recursively.

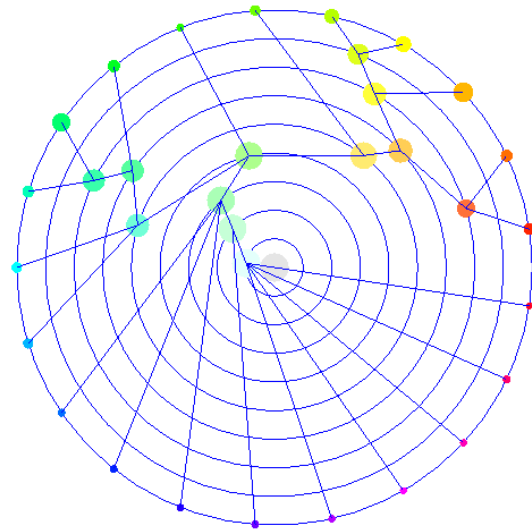


Figure 5.3: Color of hierarchical density clusters.

For drawing the nodes of the hierarchical density clusters, we use circular disks with an automatic size and color encoding. The size of the nodes are determined with respect to the size of the respective cluster that is represented by the node. We

use a logarithmic scaling to encode the size. Let r_0 be the radius of the circular disk of the root node. Then, the radius of each node N_i is determined by $r_i = r_0 \frac{\log n_i}{\log n}$, where n_i is the size of the cluster that is represented by the node N_i and n is the size of the data set.

The color of the nodes is determined with respect to the position in the radial layout. Color encoding is done using the *HSV* (Hue, Saturation, and Value) color space. Hue H encodes the angle in our radial layout and saturation S encodes the radius (distance to the origin), while value V is constant (set to 1). Hence, the applied coloring scheme can be regarded as a slice at $V = 1$ through the *HSV* color space. Figure 5.3 shows a visualization of the hierarchical density cluster. The size and color of nodes shown in this figure intuitively encode the size and the hierarchical structure of the respective clusters.

5.2 Linked views with parallel coordinates

The cluster tree visualization also serves as a user interface for interaction with linked views. We support the linked view with a parallel coordinates layout.

In parallel coordinates, we have m parallel axes and each data point is displayed as a polyline that intersects the parallel axes at the respective value of the represented attributes, where m is the number of attributes. One limitation of parallel coordinates is that they suffer from over-plotting. We display clusters by drawing a band that contains all polylines of the data points belonging to the respective cluster.

Colors are induced by the cluster tree visualization and indicate, to which cluster the drawn polyline belongs. More precisely, the colors of the multidimensional multivariate data points are assigned based on the radial layout of the hierarchical density clusters. The color for each data point is defined by the color that was assigned to the node of the cluster of the highest depth containing that data point.

When drawing the band of polylines for a cluster, this band uses the colors of the contained data points and gets assigned an opacity value that is proportional to the density of the polylines in the parallel coordinates display. The opacity is used to enhance high density and to diminish low density of polylines. More precisely, the opacity function maps a multidimensional data point to the opacity interval $[0, 1]$ by:

$$\alpha(x_i) = \left(\frac{\rho_i}{\rho_{\max}} \right)^\beta \quad (5.1)$$

where ρ_i is the density value at the multidimensional data point x_i , $\rho_{\max} = \max_{1 \leq i \leq n} \rho_i$ is the global maximum, and $\beta \geq 0$ is a scaling parameter. Figure 5.4 shows the impact of parameter β to reduce clutter in parallel coordinates: higher values of β emphasize high densities of polylines and diminish low densities of polylines.

Although the opacity function can help to reduce clutter in the parallel coordinates, it is difficult to identify clusters. Hence, we combine coloring based on the radial layout of a density cluster hierarchy and an opacity function to reduce clutter and to show clusters in parallel coordinates. User interaction is performed by

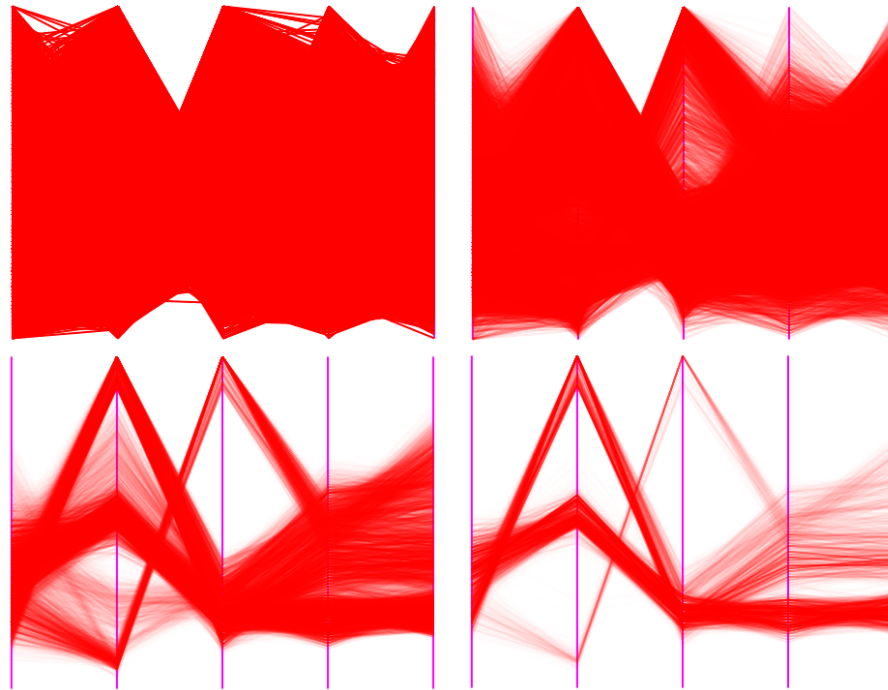


Figure 5.4: Transfer function based on density. (*Upper left*) $\beta = 0$. (*Upper right*) $\beta = 1$. (*Lower left*) $\beta = 5$. (*Lower right*) $\beta = 10$.

clicking at the clusters of interest in the cluster tree visualization.

We demonstrate the functionality of our approach by applying it to two well-known data sets. The first data set is, again, the “out5d” data set. It contains 16384 data points with five attributes (spot, magnetics, potassium, thorium, and uranium). We divide each dimension of the data set into $N = 10$ equally-sized intervals. We obtain 3661 non-empty cells and compute the hierarchical density clusters using histograms. The depth of the tree is ten, and it contains 21 leave nodes (mode clusters) and 13 internal nodes.

The hierarchical density cluster tree is displayed in Figure 5.5 (left) and its linked parallel coordinates view in Figure 5.5 (right). The cluster tree visualization exhibits an overview of the distribution of the multidimensional multivariate data, whereas parallel coordinates show clearly the values of data point attributes and their domain. The parallel coordinates allow for the exploration of individual clusters as well as the correlation between some selected clusters. Homogeneous clusters appear in a unique color in the parallel coordinate layout, while heterogeneous clusters exhibit multiple colors. Figure 5.5 (right) shows three heterogeneous clusters chosen by selecting three nodes in the cluster tree that are shown by the red dots. The cluster’s attributes are high in magnetics, low in potassium, and medium in other attributes. Moreover, the cluster partitions into three subclusters based on the attributes magnetics and uranium.

For the results in Figure 5.5, we used the hierarchical density clustering using histograms (see Section 3.2.1). We compare this to the hierarchical density clustering using kernels (see Section 3.2.2) shown in Figure 5.6. We use bandwidth parameters

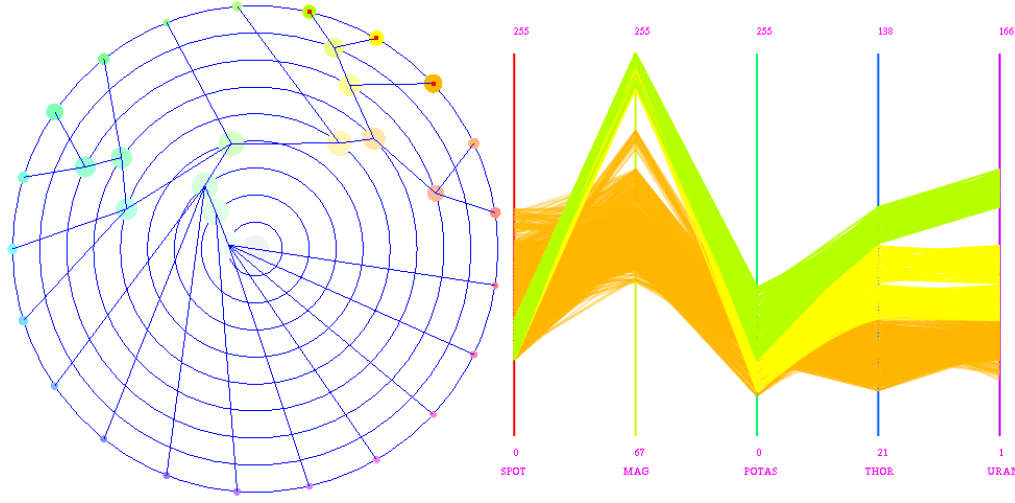


Figure 5.5: Linking hierarchical density cluster using histogram with parallel coordinates. (Left) Radial layout of the hierarchical density cluster tree. (Right) Interactively selected clusters are visualized in parallel coordinates.

$h_j = \frac{\max_j - \min_j}{N}, j = 1, \dots, 5$, where $\max_j = \max\{x_{ij}, 1 \leq i \leq n\}$, $\min_j = \min\{x_{ij}, 1 \leq i \leq n\}$, and $N = 10$. We obtain 816 number of support of points, and the hierarchical density cluster is shown in Figure 5.6 (left). The depth of the tree is four, it contains six mode clusters (one of them only contains two points). Figure 5.6 (right) shows six mode clusters that are selected and highlighted in Figure 5.6 (left).

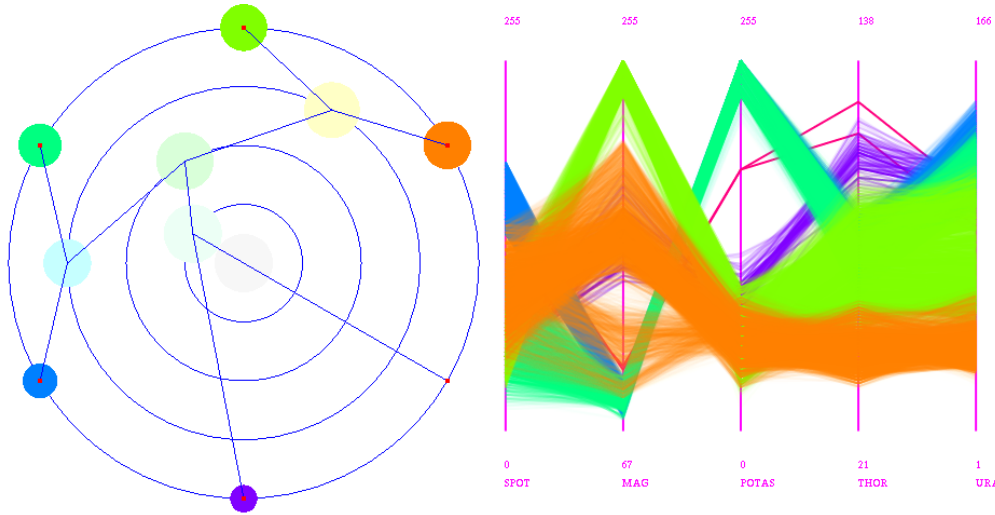


Figure 5.6: Linking hierarchical density cluster using kernel with parallel coordinates.

With the same size of cells in the hierarchical density clusters using histogram and support of points in the hierarchical density clusters using kernel, the hierarchical density cluster results are different. Some clusters in the hierarchical density clusters

using kernel can be identified in the hierarchical density clusters using histograms. The red cluster in Figure 5.6 cannot be identified in the hierarchical density clusters using histograms. If we change the size of support of points, all clusters in the hierarchical density clusters using kernel can be identified in the hierarchical density clusters using histograms. Figure 5.7 (upper) shows the hierarchical density clusters using histograms with $N = 10$. Figure 5.7 (lower) shows the hierarchical density clusters using kernel with $N = 12$. Figure 5.7 shows the same cluster results in parallel coordinates, but the hierarchical structure is not identity.

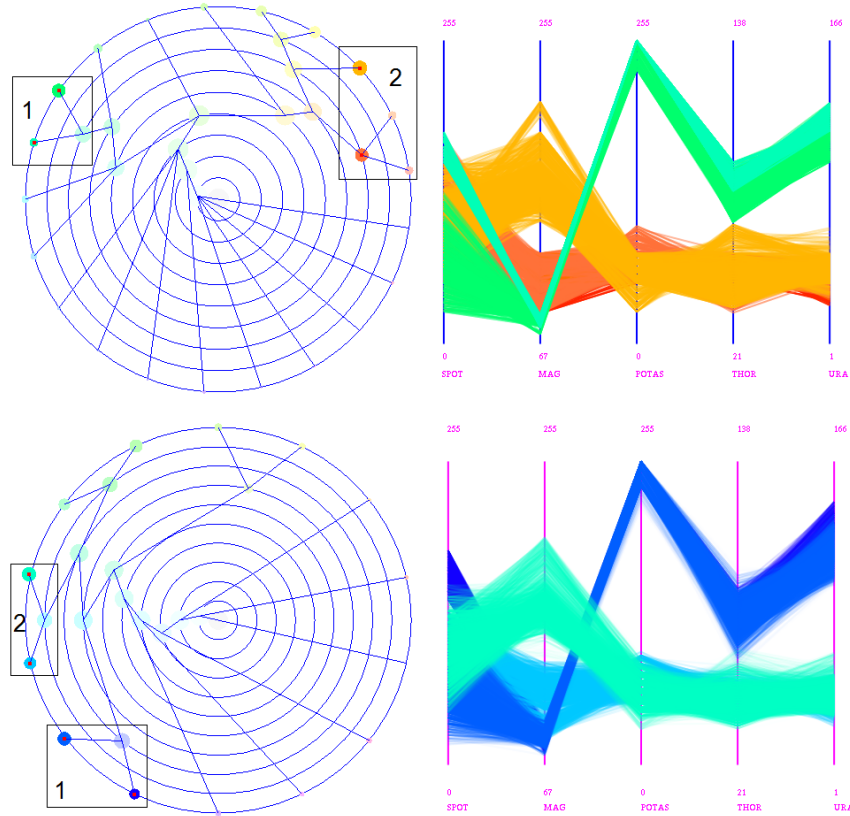


Figure 5.7: Comparison of hierarchical density cluster results using histograms and kernels. (*Upper*) Hierarchical density cluster results using histograms. (*Lower*) Hierarchical density cluster results using kernels.

Our approach can also be used for the visual analysis of multi-field spatial data sets. The feature space is a multidimensional multivariate data space, to which we apply our methods. In addition to the parallel coordinates view, we provide another linked view that shows the distribution of the data points (belonging to the selected clusters) in volumetric object space (physical space).

The second data set, we used is such a multi-field spatial data set and comes from the 2008 IEEE Visualization Design Contest [WN08]. We uniform-randomly sample the object space to obtain 1,240,000 data points with eleven feature attributes, namely total particle density, gas temperature, abundances of H mass, H_+ mass, He mass, He_+ mass, He_{++} mass, H_- mass, H_2 mass, and H_{2+} mass, and the magnitude of turbulence.

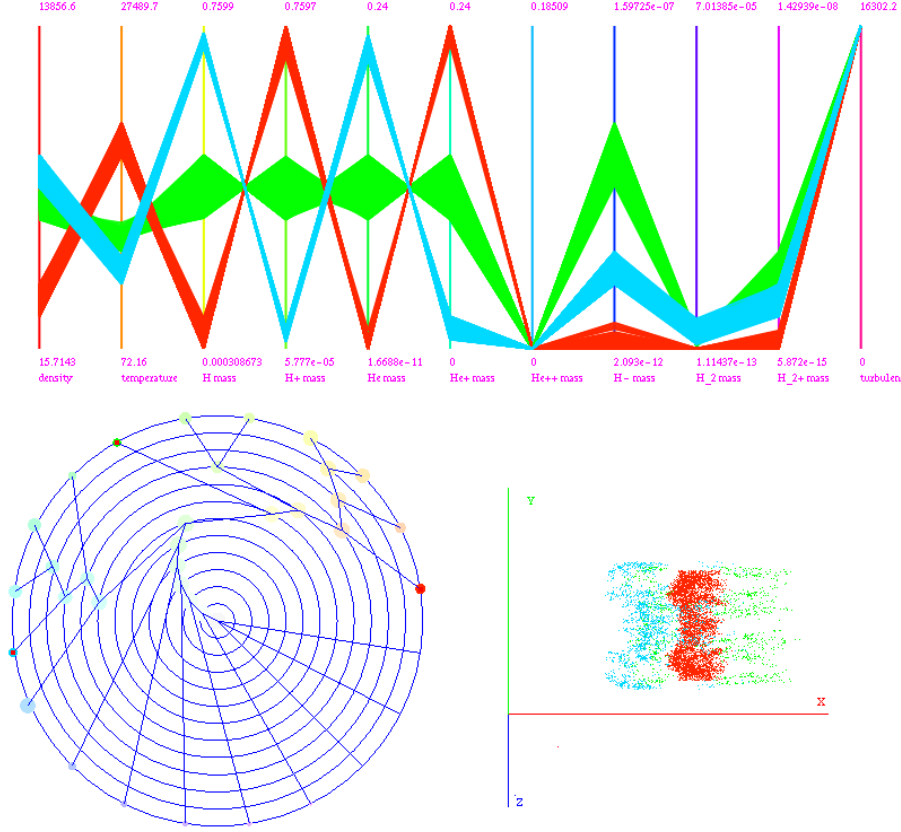


Figure 5.8: Linking cluster tree visualization with parallel coordinates and object space rendering. (*Upper*) Selected clusters in parallel coordinates (feature space). (*Left*) Radial layout of the hierarchical density cluster tree. (*Right*) Selected cluster in volumetric object space (physical space).

Figure 5.8 shows the selection of three clusters using the cluster tree interface and displaying their attributes in parallel coordinates with different colors as well as their location in physical space. The physical space rendering displays all data points that belong to the selected cluster in the respective color of the node that represents the cluster in radial layout of the hierarchical density clusters. All three selected clusters show a high magnitude of turbulence. In addition, the red cluster shows high H_+ and He_{++} mass and low H and He mass, while the blue cluster shows low H_+ and He_{++} mass and high H and He mass, and the green cluster shows medium values for H_+ , He_{++} , H , and He mass. Interestingly, in physical space the red cluster lies between the blue and green cluster, which is not true when observing the attribute values in feature space.

The linked views between 2D radial layout of the hierarchical density clusters with parallel coordinates display the hierarchical structure of density clusters and clusters on parallel coordinates. Clusters are displayed on parallel coordinates by combining color encoding and opacity. The advantage of our approach shows both homogeneous and heterogeneous clusters. In hierarchical parallel coordinates [FWR99], clusters are visualized by opacity bands, and we cannot distinguish

leaf or interior clusters of the hierarchical clustering. The opacity of polylines in parallel coordinates is similar with high-precision texture techniques [JLJC05, JLJC06], but the authors did not discuss the high-precision texture for hierarchical clustering.

5.3 Integrating circular parallel coordinates

Parallel coordinates display successfully multidimensional multivariate data, but for large data sets they suffer from clutter due to overplotting polylines and clusters. The linked view presented in the previous section avoids overplotting by selecting individual clusters. If one is interested in observing the entire data set with all (hierarchical) clusters simultaneously, one has to choose a different visualization. We propose to integrate circular parallel coordinates into our cluster tree visualization approach. The main idea is to display the attributes of each cluster in a local circular parallel coordinates system that is placed at the node positions in the radial layout. Hence, our system integrates multiple circular parallel coordinate views and hierarchical density cluster visualizations in one layout. This radial layout supports both the comprehension of the cluster distribution and a similarity/dissimilarity comparison of all clusters with respect to their attribute values.

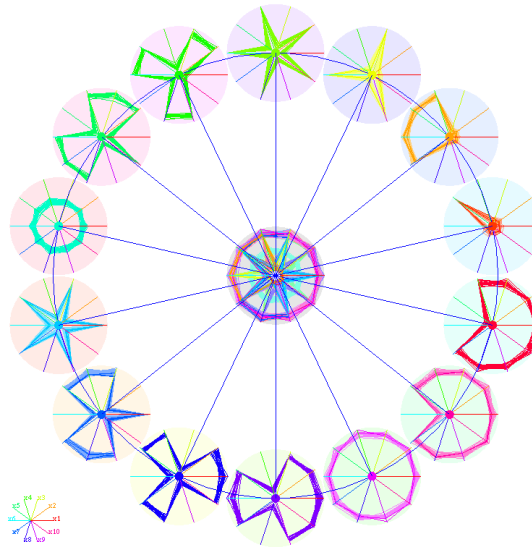


Figure 5.9: Integrated circular parallel coordinates in cluster tree visualization for data set with 14 mode clusters.

In circular parallel coordinates, the dimensions of the data sets are displayed as axes that emanate from the origin. The range of the axes is scaled and starts with its minimum values next at the origin and ends with its maximum values at the unit circle. As for standard parallel coordinates, each data point is represented as a polyline that intersects the axes at its values for the attributes. This visualization is also referred to as star glyph plots. One of the advantages of circular parallel coordinates is the efficient use of display space. To demonstrate the functionality of our visual analysis system, we apply the integrated view to a synthetic data set

containing 480 data points with ten attributes and the “out5d” data set described in Section 5.2.

Figure 5.9 shows the integrated view applied to the synthetic data set. The data set exhibits 14 clusters, which are all mode clusters. The circular parallel coordinates view in the center displays the entire data set. Because of overplotting, we cannot see how many clusters are contained and what their distribution is. Using our cluster tree visualization with integrated circular parallel coordinates, the user can easily observe the value distributions of the individual clusters and how the clusters relate to and differ from each other. Figure 1.6 shows this data set in parallel coordinates. Clusters are not visible.

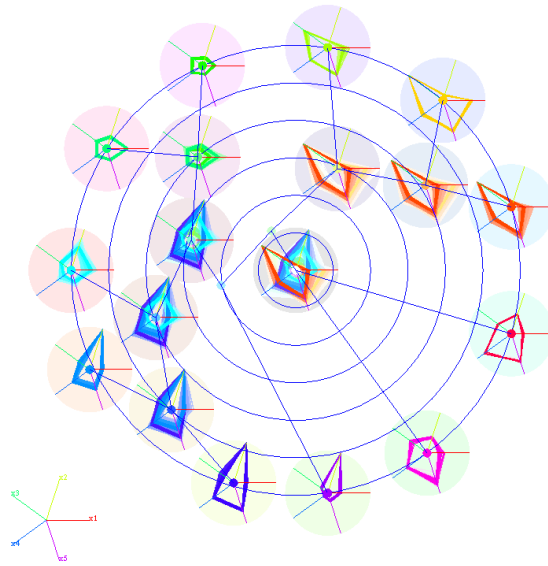


Figure 5.10: Integrated circular parallel coordinates in cluster tree visualization for data set with hierarchical clusters.

Figure 5.10 shows the integrated view applied to the “out5d” data set. This data set contains a hierarchical structure, which can be easily observed due to the cluster tree layout. Moreover, the different attribute ranges of all clusters can be investigated simultaneously.

In case of a data set with a large hierarchical structure, i.e., many clusters and high clustering depth, the available screen space for rendering each circular parallel coordinates layout may be too small to see all the details. We address this issue by providing a focus+context technique. When the user drags the cursor over the display, the current cursor position is the center of a circular focus. The focus’ size is the size of one circular parallel coordinates layout. The focus region is blown up linearly by a magnification factor, which can also be adjusted interactively. The context regions are linearly down-scaled. For the linear scaling, the current cursor position is chosen as a center and the linear scaling is applied to all rays emerging from that center. Hence, both focus and context are fully shown but at different scales. Figure 5.11 (left) shows a focus region that is expanded as in Figure 5.11 (right).

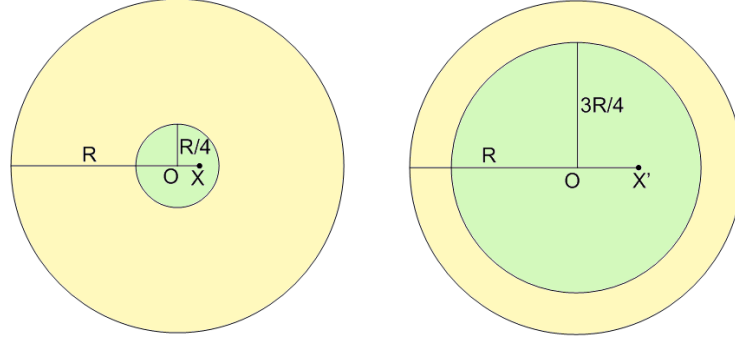


Figure 5.11: Focus + context technique. Focus region displays by green region. Context region displays by yellow region.

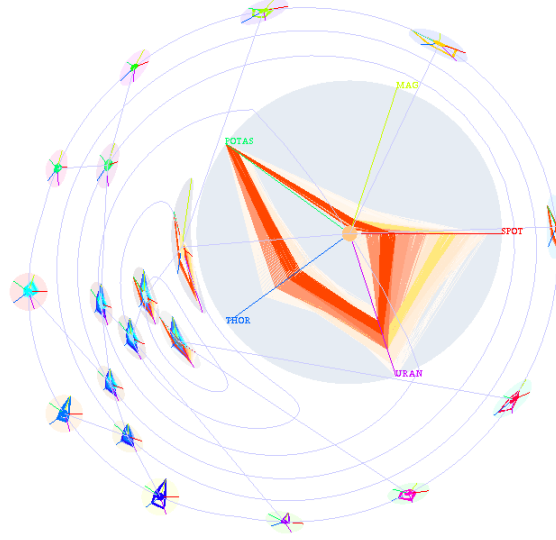


Figure 5.12: Focus+context technique for integrated visualization with cluster tree and circular parallel coordinates.

Assuming a context region within a circle with center O and radius R and a focus region within the circle with the same center O and radius $\frac{R}{4}$, the focus region is expanded to a region with center O and radius $\frac{3R}{4}$. A mapping FC from Figure 5.11 (left) to Figure 5.11 (right) is given by:

$$\overrightarrow{OX'} = FC(\overrightarrow{OX}) = \text{mag}(\|\overrightarrow{OX}\|)\overrightarrow{OX} \quad (5.2)$$

where $\text{mag}(r)$ is the magnification function,

$$\text{mag}(r) = \begin{cases} 3r & \text{if } 0 \leq r \leq \frac{R}{4}, \\ \frac{r+2R}{3} & \text{if } \frac{R}{4} \leq r \leq R \end{cases} \quad (5.3)$$

Figure 5.12 shows our integrated view with cluster tree and circular parallel coordinates when applying the focus+context technique. The cluster in the focus

can be easily investigated. It exhibits small ranges and high values for the attributes potassium and uranium, a small range and medium values for the attribute thorium, a small range and low values for the attribute magnetics, and a large range in the attribute spot. The cluster also exhibits multiple colors, which indicates that the cluster is heterogeneous.

The 2D radial layout of the hierarchical density cluster tree provides more compact views and more flexible navigation and interactive techniques than the standard dendrogram tree layout. The more compact representation allows us to assign the available screen space more efficiently when incorporating the circular parallel coordinates glyphs.

5.4 Case study

In this section, we applied our methods for visual analysis of gene expression data. A gene expression data set can be represented by a real-valued expression matrix $G = (g_{ij})_{n \times m}$ as

$$G = \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1m} \\ g_{21} & g_{22} & \cdots & g_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ g_{n1} & g_{n2} & \cdots & g_{nm} \end{pmatrix}, \quad (5.4)$$

where n is the number of genes and m is the number of samples (conditions, time steps). In the expression matrix G , the rows form the expression patterns of genes and the element g_{ij} represents the measured expression level of the i th gene in the j th sample. In gene-based clustering, genes are treated as objects while samples are treated as attributes, and the goal of gene clustering is to identify co-expressed genes that indicate co-function and co-regulation.

Euclidean distance and Pearson's correlation coefficient are common ways to identify co-expressed genes. In addition, Euclidean distance can be more appropriate for log-ratio data, while Pearson's correlation coefficient can be better for absolute-valued data.

Euclidean distance Before using Euclidean distance of genes, each gene is standardized with zero mean and variance one. For each gene $g_i = (g_{i1}, \dots, g_{im})$ the mean value μ_i and the variance σ_i are given by:

$$\begin{aligned} \mu_i &= \frac{\sum_{k=1}^m g_{ik}}{m}, \\ \sigma_i &= \sqrt{\frac{\sum_{k=1}^m (g_{ik} - \mu_i)^2}{m}}. \end{aligned} \quad (5.5)$$

Therefore, gene g_i is transformed to $\tilde{g}_i = (\tilde{g}_{i1}, \dots, \tilde{g}_{im})$ with

$$\tilde{g}_{ik} = \frac{g_{ik} - \mu_i}{\sigma_i}, k = 1, \dots, m.$$

Without loss of generality, we assume that each gene g_i is standardized with zero mean and variance one such that

$$\|g_i\| = \sqrt{\sum_{k=1}^m g_{ik}^2} = \sqrt{m}.$$

Hence, all genes lie on the surface of a hypersphere with radius $R = \sqrt{m}$, and the Euclidean distance is given by

$$\|g_i - g_j\| = R\sqrt{2(1 - \cos(g_i, g_j))}.$$

Pearson's correlation coefficient Pearson's correlation coefficient between two genes g_i and g_j is defined as

$$\rho(g_i, g_j) = \frac{\sum_{k=1}^m (g_{ik} - \mu_i)(g_{jk} - \mu_j)}{\sqrt{\sum_{k=1}^m (g_{ik} - \mu_i)^2} \sqrt{\sum_{k=1}^m (g_{jk} - \mu_j)^2}}. \quad (5.6)$$

We transform gene g_i to $\tilde{g}_i = (\tilde{g}_{i1}, \dots, \tilde{g}_{im})$ with

$$\tilde{g}_{ik} = \frac{g_{ik} - \mu_i}{\sqrt{\sum_{k=1}^m (g_{ik} - \mu_i)^2}}, k = 1, \dots, m.$$

We have $\|\tilde{g}_i\| = 1$ and $\rho(g_i, g_j) = \rho(\tilde{g}_i, \tilde{g}_j) = \sum_{k=1}^m \tilde{g}_{ik}\tilde{g}_{jk} = \cos(\tilde{g}_i, \tilde{g}_j)$.

In both cases, we assume that the gene expression data lie on a unit hypersphere. Moreover, Euclidean distance and Pearson's correlation coefficient relate to each other as follows

$$\|g_i - g_j\|^2 = 2(1 - \cos(g_i, g_j)) = 2(1 - \rho(g_i, g_j)). \quad (5.7)$$

We apply the hierarchical density clustering using kernel algorithm in Section 3.2.2. We define that two genes g_i and g_j are co-expressed if and only if $\rho(g_i, g_j) \geq \rho_0$, where ρ_0 is a threshold parameter. This is equivalent with $\|g_i - g_j\| \leq r_0$, where $r_0 = \sqrt{2(1 - \rho_0)}$ as a threshold distance. First, for each gene g_i we find all genes that are co-expressed with g_i and generate support of points. Second, two support of points $B(g_i, r_0)$ and $B(g_j, r_0)$ are intersecting if $\cos(g_i, g_j) \geq \cos(2(\arccos \rho_0))$ or $\|g_i - g_j\| \leq r_0\sqrt{4 - r_0^2}$, respectively.

The gene expression data set, we are using is called the ‘‘Serum’’ data set. The data set contains 2,467 genes, each of which is described by twelve time steps. We set the threshold parameter $\rho_0 = 0.90$. The hierarchical density cluster tree of the data set is presented in Figure 5.13 (left), and Figure 5.13 (right) displays all six

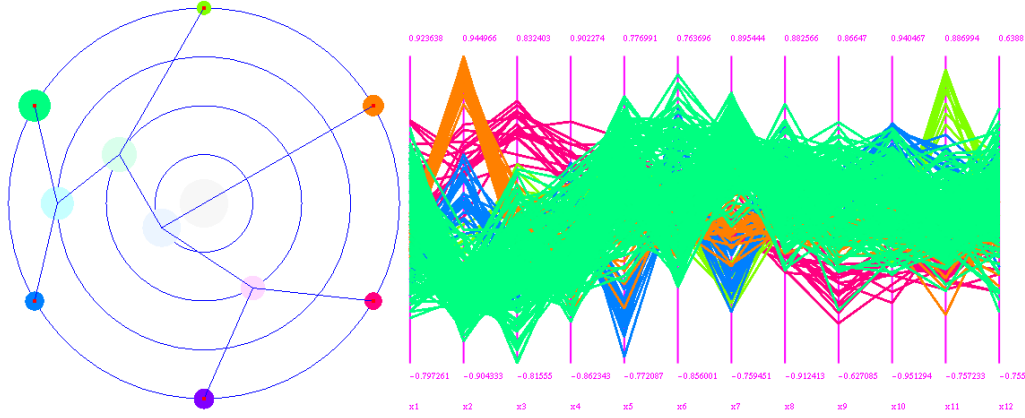


Figure 5.13: Linked views “Serum” data set. (Left) Hierarchical density clusters. (Right) All mode clusters display on parallel coordinates.

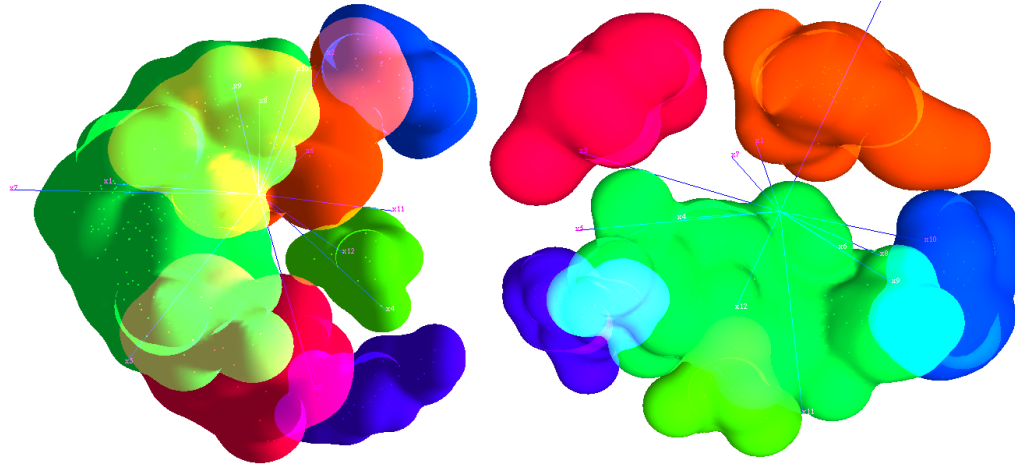


Figure 5.14: Visualizing “Serum” data set with optimized 3D star coordinates. Mode clusters are well separated. Two views are shown.

mode clusters using linked parallel coordinates. The mode clusters displayed on the parallel coordinates are very cluttered. Figure 5.14 shows the mode clusters with 3D optimized star coordinates. The mode clusters are well separated.

The other gene expression data set, we are using is called the “Yeast cell cycle” data set in [CCW⁺98]. The data set contains 6600 genes over 17 time steps. This data set measure absolute values. The threshold parameter is chosen for this data set $\rho_0 = 0.86$. The hierarchical density cluster tree is shown in Figure 5.15 (left). It contains five homogeneous clusters. A node in the hierarchical density cluster tree is selected. The homogeneous cluster corresponding to this node is represented in parallel coordinates in Figure 5.15 (right). All genes in the cluster are similar patterns. Figure 5.16 (right) shows all homogeneous clusters in optimized 3D star coordinates. The clusters are well separated in visual space. The hierarchical density clusters using histogram cannot apply for gene expression data set. The gene expression data is transformed before clustering. All genes are placed in a small

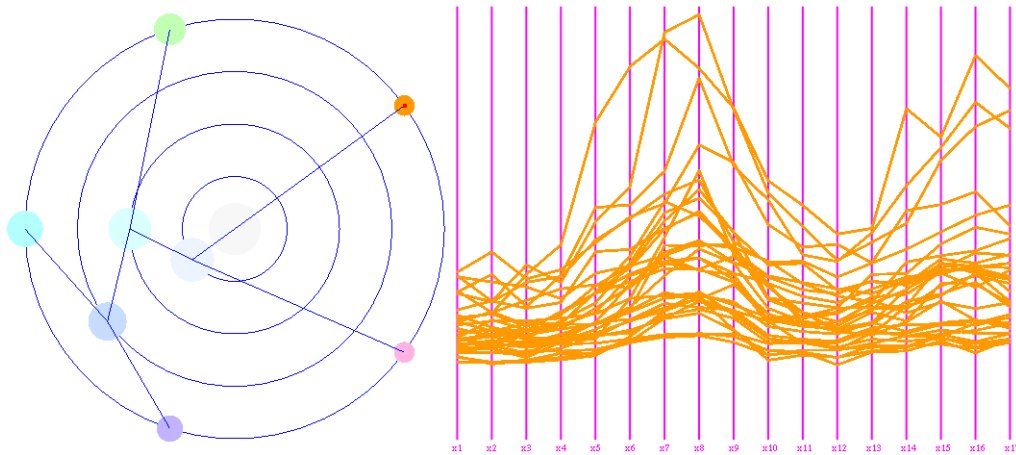


Figure 5.15: Linked views “Yeast cell cycle” data set. (*Left*) Hierarchical density clusters. (*Right*) A selected mode clusters display on parallel coordinates.

cone region of a unit hyper-sphere.

Our system is helpful for biologists. Our system can automatically identify clusters of genes, which express similar patterns in the data. Clustering results are visualized in our system. In the radial layout of the hierarchical density clusters, researchers get an overview of the distribution of the clusters. If researchers select a cluster on the radial layout view, all genes in the cluster are shown in the parallel coordinates view. In the parallel coordinates view, researchers can see the pattern of the cluster, i.e., the cluster has high or low expression levels over all samples, as shown in Figure 5.15. To compare the correlation and differences between two

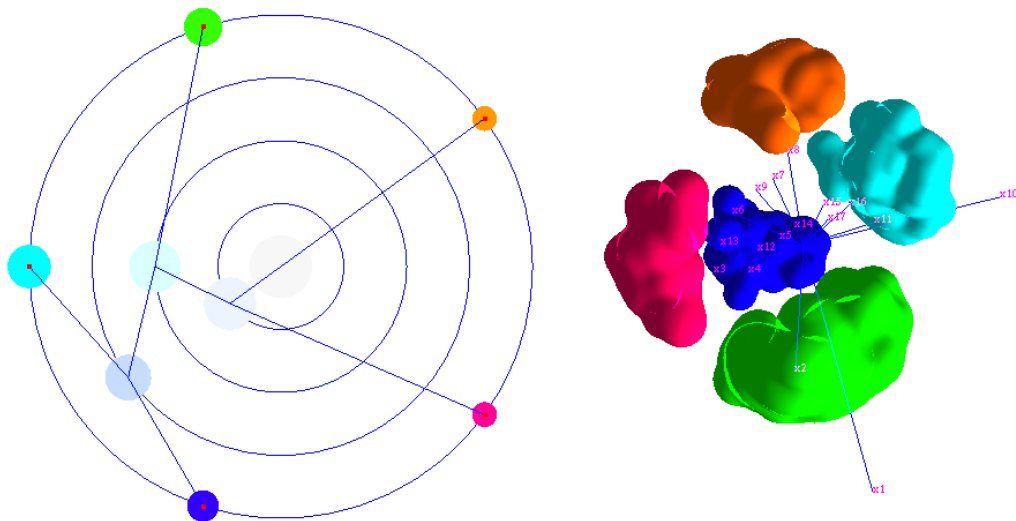


Figure 5.16: Visualizing “Yeast cell cycle” data set with optimized 3D star coordinates. Mode clusters are well separated.

clusters, researchers can select the two clusters using the radial layout. All genes in the two clusters are shown in parallel coordinates view and researchers can see

the different expression levels of the two clusters over all samples. Because of the overplotting in parallel coordinates, many clusters shown simultaneously in parallel coordinates does not support to visual comparison of the difference between these clusters. Figure 5.13 shows five highly cluttered of clusters. The optimized star coordinates can simultaneously visualize multiple clusters. Figure 5.14 and Figure 5.16 show five clusters. In the star coordinates view, researchers also see the pattern of clusters. However, it shows the expression level less accurately than the parallel coordinates view, as the projection introduces ambiguity.

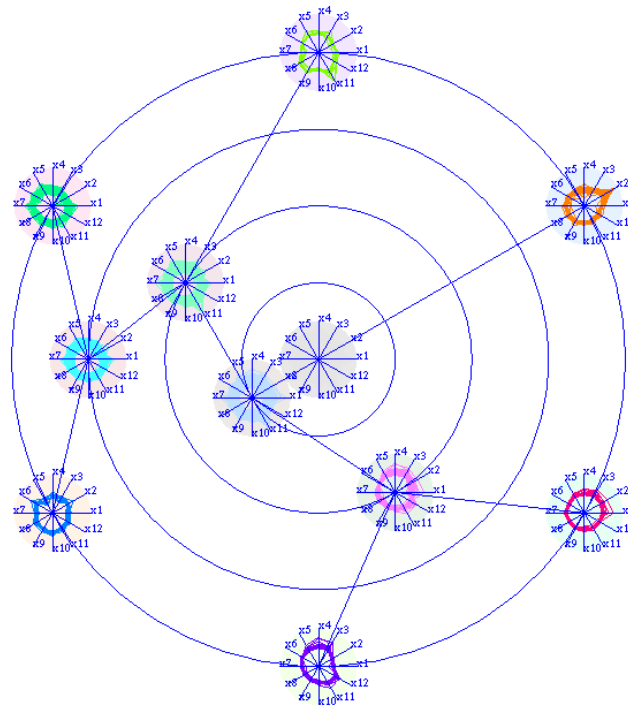


Figure 5.17: Visualizing the hierarchy of co-expressed gene clusters in “Serum” data set.

The dendrogram is a traditional method for visualizing gene expression data set. The dendrogram is representing the hierarchical clustering using single linkage clustering. For large-scale gene expression data, the dendrogram is very complex and clusters are not identified automatically [SS02]. In our approach, the hierarchical structure is reduced and clusters are automatically determined. Users can select nodes in radial layout of the hierarchical density clusters and all genes in these clusters are displayed in parallel coordinates or star coordinates. Figure 5.17 shows the hierarchy of co-expressed gene clusters in the “Serum” data set. The integrated circular parallel coordinates into 2D radial layout of the hierarchical density clusters supports to display the hierarchy of clusters simultaneously.

For gene expression data, heatmaps are more common than the use of parallel coordinates. To provide the display that is familiar to the biologists, we replace the circular parallel coordinates by a heatmap. The heatmap is also embedded in the 2D radial layout. However, the heat map does not display the individual genes

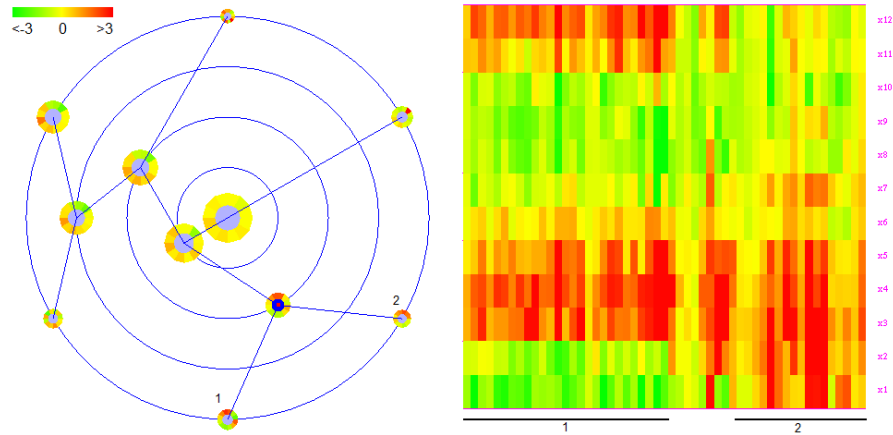


Figure 5.18: Linked heatmap view for the “Serum” data set. (*Left*) Hierarchical density clusters with embedded heatmaps for the means. (*Right*) A cluster is displayed with all genes and attributes in the heatmap view. The two subclusters marked as “1” and “2” can easily be observed. Genes belonging to the selected cluster but not to any of the two subclusters are displayed in between.

but the mean of clusters. Figure 5.18 (left) shows the mean of the clusters of the “Serum” data set. The values of the genes are mapped into colors. The color scale range that is used is the one that is familiar to the biologists. It maps the hues green to yellow to log ratios -3 to 0 and the hues yellow to red to log ratios 0 to 3 . The advantage of color encoding the mean clusters is that it shows the difference between clusters. Showing the heatmap for all genes within the clusters would not scale well. Therefore, we support a linked view, in addition, where the heatmap of all genes within a selected cluster is shown. Figure 5.18 (right) shows a linked view of the 2D radial layout with a heatmap view. In the heatmap, genes in homogeneous clusters (leaf nodes) are ordered randomly and genes in heterogeneous clusters (internal nodes) are ordered by inorder traversal of the hierarchical density cluster tree. In Figure 5.18 (left), a cluster (internal node) is selected in the 2D radial layout. All genes within the cluster are displayed in the heatmap view in Figure 5.18 (right). We can easily identify the two subclusters within the selected cluster. They appear to the left and the right of the heatmap, respectively. Genes belonging to the selected cluster but not belonging to the two subclusters, are displayed between the two subclusters. The example also shows that the random order within a homogeneous cluster is a valid choice, as the attributes exhibit a high similarity.

Figure 5.19 shows the hierarchical clustering structure of the “Serum” data set using the Hierarchical Clustering Explorer (HCE) version 3.5 [SS02]. When the similarity bar indicates value 0.537, one gets six clusters. All clusters are shown in the heatmap and linked views with profiles. Comparing visually with Figure 5.19, clusters can more intuitively be observed in Figure 5.17 and the gene expression level of clusters are also identified more intuitively using embedded circular parallel coordinates than using heatmaps. We made a user study with ten participants to compare visual explorations using the two systems. We refer to the systems as

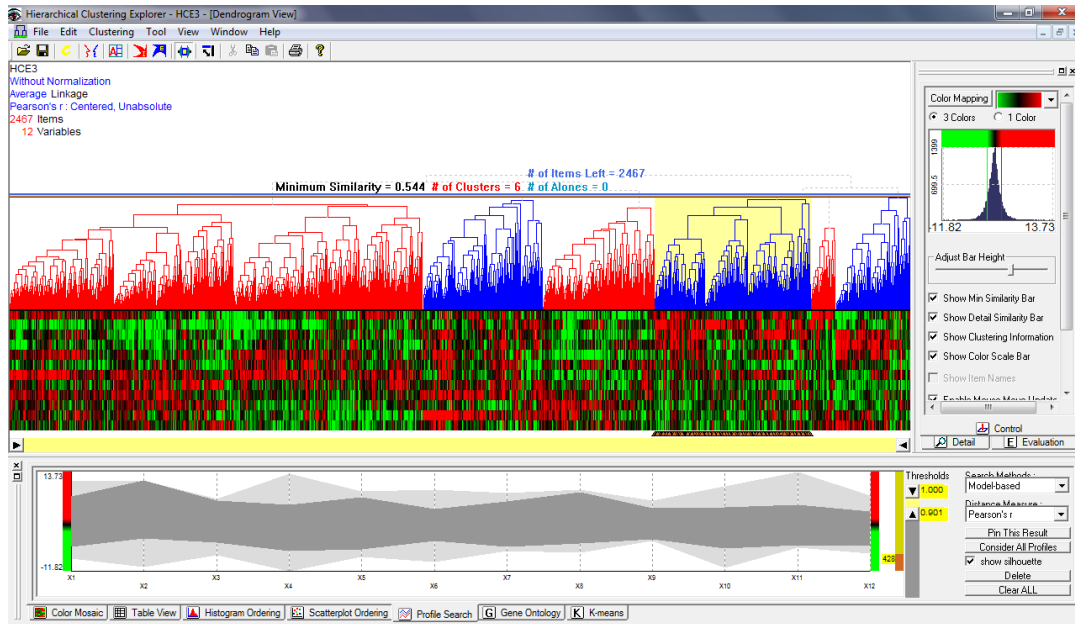


Figure 5.19: Hierarchical Clustering Explorer of the “Serum” data set.

HCE and CPC (Circular Parallel Coordinates). We asked the test persons four questions, checked the correctness of the results, and measured the time it took them to complete the tasks. The questions were the same for both systems. Half of the test persons first fulfilled the tasks using the HCE system and afterwards using the CPC system. The other half did it vice versa. The four exploration tasks were specified by the following four questions:

1. What is the average variability within a selected group?
2. Which attribute has the highest variability within the group?
3. How many attributes have a variability larger than 20 %?
4. What other group is the most similar to the group?

	HCE	CPC	HCE	CPC
1	20 %	30 %	20.5 (s)	12.8 (s)
2	80 %	100%	10.0 (s)	6.4 (s)
3	60%	50%	11.2 (s)	13.5 (s)
4	30%	20%	45.8 (s)	19.2 (s)

Table 5.1: Comparison between HCE and CPC on gene expression data: Correctness (in percent) and timings (in seconds).

Table 5.1 presents percentage of the right answers and average duration to give the answers (in seconds). For the HCE system, we first tried to provide the test persons

with the heatmaps and the dendograms only. They were not able to give an answer to Questions 1-3. So, we also provided them with the linked parallel coordinates, which basically reduced the comparison to comparing standard parallel coordinates with circular parallel coordinates. As expected, there was no obvious preference that could be documented for the results of Questions 1-3. For Question 4, one needs to compare different clusters, i.e., it goes beyond just looking at a single cluster. When asking for the cluster closest to the blue mode cluster in Figure 5.17, all test persons would immediately find the right solution (the green mode cluster), as this is indicated by the cluster hierarchy. So, for the numbers given in Table 5.1, we made the task more difficult and asked for the mode cluster closest to the orange one. The percentage of right answers in HCE and CPC are rather low and exhibit a slight difference only the duration to answer the questions using HCE are significantly longer than when using CPC. The interpretation of the user study would be that heatmaps are significantly harder to evaluate quantitatively than parallel coordinates and that linked parallel coordinates require significant user interaction, which can be avoided using embedded circular parallel coordinates. Moreover, embedded circular parallel coordinates provide an intuitive means for visual data exploration tasks.

Chapter 6

Conclusion and future work

We have presented a framework for visual analysis of multidimensional multivariate data based on hierarchical cluster visualization. Our system incorporated automatic computations of hierarchical density clusters using efficient grid-based algorithms, visualization of the hierarchical density cluster tree using an optimized star coordinates with a nested level set visualization for density clusters, a 2D radial layout with linked views to parallel coordinates rendering, and the integration of circular parallel coordinates into the radial layout cluster tree layout.

Chapter 3 introduced two algorithms to create hierarchical density clusters. First, a hierarchical density cluster was created by a top-down approach based on histogram density estimation. The traditional histogram density estimation was simple, but it was not efficient for high dimensional space. By combining it with a new partitioning in high dimensional space, we developed a method that is efficient and scalable with both the size and the dimensions of the data set. Second, a hierarchical density cluster was created by a bottom-up approach based on kernel density estimation. We proposed another effective method for partitioning high-dimensional space and a more accurate estimation of density in which the data set is partitioned into support of points (intersecting partition). The advantages of the two algorithms were fast, capable any shapes of density clusters, and can handle noise of the data sets.

Chapter 4 presented a method for visualizing hierarchical density clusters based on optimized star coordinates. The optimized star coordinate system was defined such that it maintains distances between two barycenters of mode clusters when high-dimensional data was projected into visual space. We introduced both 2D and 3D optimized star coordinates. In 2D optimized star coordinates, clusters were visualized by enclosing contours, and in 3D optimized star coordinates, clusters were visualized by enclosing surfaces. A nested level set visualization for the high density area with respect to different density levels allowed for an interactive exploration of the hierarchical clusters and to correlate the clusters to the original dimensions.

Chapter 5 presented a system for the visual analysis of multidimensional multivariate data based on hierarchical cluster visualization. Our system incorporated visualization of the hierarchical density cluster tree using a 2D radial layout, linked views to parallel coordinates and object space renderings, and the integration of circular parallel coordinates into the radial cluster tree layout. The 2D radial lay-

out of the hierarchical density cluster tree supported an intuitive visualization to summarize the distribution structure of the data set (clusters with different density levels). The colors were assigned automatically by mapping the *HSV* color space to the radial layout and allowed for intuitive linking. The combination of color and opacity supported an intuitive visualization of selected clusters in a linked parallel coordinates view. The integration of circular parallel coordinates can solve the overplotting problem for large data by displaying clusters in multiple views embedded into the cluster tree layout. The linked object-space view was important in the context of spatial multi-channel data.

There are some limitations of the presented approaches that are left for future work.

- As the hierarchical density clusters are based on continuous density distribution of multidimensional data sets, the clustering algorithms do not handle data set that contain category attributes.
- The density distribution of multidimensional data is estimated based on histogram or kernel methods, which depend on some parameters (grid sizes in histogram method and kernel sizes in kernel method). The clustering algorithms are not automatically selecting these parameters.
- The hierarchical density clusters are created in full dimensions of the data sets. Therefore, the clustering algorithms are not applied for high dimensionality, because clusters may not exist with full dimensions.
- When visualizing hierarchical density clusters, the optimized star coordinates only consider the homogeneous clusters, that do not consider the hierarchical structure of density clusters.
- Due to our design goal to use linear projection only, clusters that are well-separated in the multidimensional space may still overlap in visual space.
- The radial layout of the hierarchical density clusters did not support the similarity between clusters.
- Our system is capable to handle multidimensional data of up to 20 or even 50 dimensions. A future direction would be to look into data sets with higher dimensionality, i.e., having hundreds or thousands of dimensions. A common way to overcome the problem of high-dimensional data is reducing its dimensionality. Feature transformations are commonly used on high-dimensional data sets that include techniques such as principal component analysis (PCA) and singular value decomposition (SVD). Hence, high-dimensional data is transformed into a lower-dimensional space of up to 50 dimensions. Another common technique of reducing dimensions is feature selection, that selects a subset of relevant dimensions and removes irrelevant dimensions. In high-dimensional data sets, clusters can only exist in subspaces. Hierarchical density clusters can be developed to identify clusters in a subspace, i.e., the hierarchical density clusters can handle both hierarchical dimensions and hierarchical

clusters. Hence, a system will need to be developed to handle this situation. Some specific application areas of high-dimensional data cluster analysis can be considered such as gene expression data analysis or text documents.

As a consequence, with the advantage of visualizing hierarchical density clusters was supported both an overview and different levels of detail of the multidimensional multivariate data sets. We believe the application of our system will be fruitful.

References

- [ABK98] Mihael Ankerst, Stefan Berchtold, and Daniel A. Keim. Similarity clustering of dimensions for an enhanced visualization of multidimensional data. In *Proc. of IEEE symposium on Information Visualization*, pages 52–60, 1998.
- [ABKS99] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Joerg Sander. Optics: Ordering points to identify the cluster structure. In *ACM SIGMOD international conference on Management of data*, pages 49–60, 1999.
- [AC91] Bowen Alpern and Larry Carter. The hyperbox. In *Proceeding of the 2nd conference on Visualization*, pages 133–139, 1991.
- [AdO04] Almir O. Artero and Maria C. F. de Olivira. Viz3d: Effective exploratory visualization of large multidimensional data sets. *Computer Graphics and Image Processing, 17th Brazilian Symposium on SIB-GRAPI*, pages 340–347, 2004.
- [AdOL04] Almir O. Artero, Maria C. F. de Oliveira, and Haim Levkowitz. Uncovering clusters in crowded parallel coordinates visualizations. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 81–88, 2004.
- [AdOL06] Almir O. Artero, Maria C. F. de Oliveira, and Haim Levkowitz. Enhanced high dimensional data visualization through dimension reduction and attribute arrangement. In *Information Visualization 2006*, pages 707–712, 2006.
- [AKK96] Mihael Ankerst, Daniel A. Keim, and Hans-Peter Kriegel. Circle segments: A technique for visually exploring large multidimensional data sets. In *Proceedings of Visualization (Hot Topics Session)*, 1996.
- [And72] D. F. Andrews. Plots of high-dimensional data. *Biometrics*, 28:125–136, 1972.
- [BAS05] Enrico Bertini, Luigi Dell’ Aquila, and Giuseppe Santucci. Springview: Cooperation of radviz and parallel coordinates for view optimization and clutter reduction. *Proceedings of the Coordinated and Multiple Views in Exploratory Visualization*, pages 22–29, 2005.

- [Bed90] Jeff Beddow. Shape coding of multidimensional data on a microcomputer display. In *Proceedings of the 1st conference on Visualization*, pages 238–246, 1990.
- [BG05] Ingwer Borg and Patrick J. F. Groenen. *Modern Multidimensional Scaling: Theory and Applications (second edition)*. Springer, New York, 2005.
- [Bli82] James F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):542–547, July 1982.
- [CB07] Nathan D. Coorprider and Robert P. Burton. Extension of star coordinates into three dimensions. In *Proceedings of the SPIE*, volume 6495, 2007.
- [CCW⁺98] Raymond Cho, Michael Campbell, Elizabeth Winzeler, Lars Steimetz, Andrew Conway, Lisa Wodicka, Tyra Wolfsberg, Andrei Gabrielian, David Landsman, David Lockhart, and Ronald Davis. A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2(1):65–73, July 1998.
- [Che73] Herman Chernoff. The use of faces to represent points in k -dimensional space graphically. *Journal of the Maerican Statistical Association*, 68(3):361–368, 1973.
- [CK06] Robert Chud and Jaroslav Kadlec. Foxi-hierarchical structure visualization. In *Advances in Systems, Computing Sciences and Software Engineering*, pages 229–233, 2006.
- [CL04] Keke Chen and Ling Liu. Clustormap: Labeling clusters in large datasets via visualization. In *Proceedings og the 13th ACM International conference on information and knowledge managenment*, pages 285–293, 2004.
- [Cle93] William S. Cleveland. *Visualizing Data*. Hobart Press, Summit, New Jersey, 1993.
- [CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2001.
- [CMS99] Stuart K. Card, Jock Mackinlay, and Ben Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufman, 1999.
- [dBCvKO08] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications (3rd edition)*. Springer, 2008.

- [DMS98] Inderjit S. Dhillon, Dharmendra S. Modha, and W. Scott Spangler. Visualizing class structure of multidimensional data. *Proceedings of the 30th Symposium on the Interface: Computing Science and Statistics*, 41:488–493, 1998.
- [DMS02] Inderjit S. Dhillon, Dharmendra S. Modha, and W. Scott Spangler. Class visualization of high-dimensional data with applications. *Computational Statistics and Data Analysis*, pages 59–90, 2002.
- [EKSX96] Martin Ester, Hans-Peter Kriegel, Joerg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the second international conference on knowledge discovery and data mining*, pages 226–231, 1996.
- [EW93] Stephen G. Eick and Graham J. Wills. Navigating large networks with hierarchies. In *proceedings of the 4th conference on Visualization*, pages 204–209, 1993.
- [FB90] Steven Feiner and Clifford Beshers. Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds. In *Proceedings of the 3rd annual ACM SIGGRAPH symposium on User interface software and technology*, pages 76–83, 1990.
- [FB94] George W. Furnas and Andreas Buja. Prosection views: dimensional inference through sections and projections. *Journal of Computational and Graphical Statistics*, 3(4):323–385, 1994.
- [FGW02] Usama Fayyay, Geoges Grinstein, and Andreas Wierse. *Information Visualization in Data mining and Knowledge Discovery*. Morgan Kaufman, 2002.
- [FT74] Jerome H. Friedman and John Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, 23(9):881–890, 1974.
- [FWR99] Ying-Huey Fua, Matthew O. Ward, and Elke A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. *Proceedings of IEEE Symposium on Information Visualization*, pages 43–50, 1999.
- [GK03] Martin Graham and Jessie Kennedy. Using curves to enhance parallel coordinate visualizations. In *Information Visualization 2003, Proc. Seventh International conference*, pages 10–16, 2003.
- [GKWZ07] Alexander N. Gorban, Balazs Kegl, Donald C. Wunsch, and Andrei Zinovyev. *Principal manifolds for data visualization and dimension reduction*. Springer, Berlin-Heidenberg-New York, 2007.

- [GRS98] Sudipto Guha, Rajeev Rastogi, and Kyusoek Shim. Cure: An efficient clustering algorithm for large databases. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 73–84, 1998.
- [Har75] John A. Hartigan. *Clustering Algorithm*. Wiley, 1975.
- [Har81] John A. Hartigan. Consistency of single linkage for high-density clusters. *Journal of the American Statistical Association*, 76(374):388–394, 1981.
- [HDWB95] Robert J. Hendley, Nick S. Drew, Andy M. Wood, and Russell E. Beale. Narcissus: Visualizing information. In *IEEE symposium on Information Visualization*, pages 90–96, 1995.
- [HGM⁺97] Patrick Hoffman, Georges Grinstein, Kennedth Marx, Ivo Grosse, and Eugene Stanley. Dna visual and analytic data mining. *Proceedings of the 8th conference on Visualization*, pages 437–441, 1997.
- [HK98] Alexander Hinneburg and Daniel Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the fourth international conference on knowledge discovery and data mining*, pages 58–65, 1998.
- [HK03] Alexander Hinneburg and Daniel Keim. A general approach to clustering in large databases with noise. In *Knowledge and Information Systems*, volume 5, pages 387–415, 2003.
- [HK06] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2006.
- [HLD02] Helwig Hauser, Florian Ledermann, and Helmut Doleisch. Angular brushing of extended parallel coordinates. In *Proc. Symposium Information visualization*, pages 127–130, 2002.
- [Hub85] Peter J. Huber. Projection pursuit. *The Annals of Statistics*, 13(2):435–475, Jun. 1985.
- [ID90] Alfred Inselberg and Bernard Dimsdale. Parallel coordinates: A tool for visualizing multidimensional geometry. In *Proceedings of the 1st conference on Visualization*, pages 361–378, 1990.
- [Ins85] Alfred Inselberg. The plane with parallel coordinates. *Visual Computer*, 1:69–97, 1985.
- [JLJC05] Jimmy Johansson, Patric Ljung, Mikael Jern, and Matthew Cooper. Revealing structure within clustered parallel coordinates displays. *Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, pages 125–132, 2005.

- [JLJC06] Jimmy Johansson, Patric Ljung, Mikael Jern, and Matthew Cooper. Revealing structure in visualizations of dense 2d and 3d parallel coordinates. *Information Visualization*, 5(2):125–136, 2006.
- [Joh93] Brian Scott Johnson. *Treemaps: Visualizing hierarchical and categorical data*. PhD thesis, Department of Computer Science, University of Maryland, 1993.
- [Jol86] Ian T. Jolliffe. *Principal Component Anylsis (second edition)*. Springer Verlag, 1986.
- [JS91] Brian Johnson and Ben Shneiderman. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the 2nd conference of Visualization '91*, pages 284–291, 1991.
- [JTJ04] Jimmy Johansson, Robert Treloar, and Mikael Jern. Integration of unsupervised clustering, interaction and parallel coordinates for the exploration of large multivariate data. *Proceedings of the Information Visualisation*, pages 52–57, 2004.
- [Kan00] Eser Kandogan. Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. *Proceedings of IEEE information Visualization Symposium (Hot Topics)*, pages 4–8, 2000.
- [Kan01] Eser Kandogan. Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. *Proc. ACM Int. Conf. Knowledge Discovery and Data Mining*, pages 107–116, 2001.
- [Kei97] Daniel A. Keim. Visual techniques for exploring databases. In *Invited tutorial, Int. Conference on Knowledge Discovery in Databases, KDD97*, 1997.
- [Kei02] Daniel A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 7(1):100–107, January-March 2002.
- [KHK99] George Karypis, Eui Hong Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, August 1999.
- [KK94] Daniel A. Keim and Hans-Peter Kriegel. Visdb: Database exploration using multidimensional visualization. *Computer Graphics and Applications*, pages 40–49, 1994.
- [KKA95] Daniel A. Keim, Hans-Peter Kriegel, and Mihael Ankerst. Recursive pattern: A technique for visualizing very large amounts of data. In *Proceedings of Visualization*, pages 279–286, 1995.
- [Koh95] Teuvo Kohonen. *Self-Organizing Maps (third edition)*. Springer, 1995.

- [Lev91] Haim Levkowitz. Color icons: Merging color and texture perception for integrated visualization of multiple parameters. In *Proceeding of the 2nd conference on Visualization*, pages 22–25, 1991.
- [LL09a] Tran Van Long and Lars Linsen. Multiclustertree: Interactive visual exploration of hierarchical clusters in multidimensional multivariate data. In *Eurographics IEEE-VGTC Symposium on Visualization 2009*, volume 28, pages 823–830, 2009.
- [LL09b] Tran Van Long and Lars Linsen. Visualizing high density cluster in multidimensional data using optimized star coordinates. *Computational Statistics (submitted)*, 2009.
- [LLR09] Lars Linsen, Tran Van Long, and Paul Rosenthal. Linking multi-dimensional feature space cluster visualization to multifield surface extraction. *IEEE Computer Graphics and Applications*, 29(3):85–89, 2009.
- [LLRR08] Lars Linsen, Tran Van Long, Paul Rosenthal, and Stephan Rosswoog. Surface extraction from multi-field particle volume data using multi-dimensional cluster visualization. In *IEEE Transactions on Visualization and Computer Graphics*, volume 14, pages 1483–1490, 2008.
- [LRP95] John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the ACM Conference on Human Factors and Computing Systems*, pages 401–408, 1995.
- [LWW90] Jeffrey LeBlanc, Matthew O. Ward, and Norman Wittels. Exploring n-dimensional databases. In *Proceedings of the 1st conference on Visualization*, pages 230–239, 1990.
- [MGTS90] Ted W. Mihalisin, E. Gawlinski, John Timlin, and John Schwegler. Visualizing a scalar field on an n -dimensional lattice. In *Proceeding of the 2nd conference on Visualization*, pages 255–262, 1990.
- [MW06] Rida E. A. Moustafa and Edward J. Wegman. Multivariate continuous data - parallel coordinates. In *Graphics of Large datasets: Visualizing a Million*, Springer, pages 143–256, 2006.
- [Nh06] Matej Novotny and Helwig hauser. Outlier-preserving focus+context visualization in parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):893–900, 2006.
- [NIS03] NIST/SEMATECH. *e-Handbook of Statistical Methods*. <http://www.itl.nist.gov/div898/handbook/>, 2003.
- [PG88] Ronald M. Pickett and Georges G. Grinstein. Iconographic displays for visualizing multidimensional data. In *Proceedings Conference on Systems, Man and Cybernetics*, pages 514–519, 1988.

- [Pic70] Ronald M. Pickett. Visual analyses of texture in the detection and rrecognition of objects. In *in Picture Processing and Psychopictorics*, B. S. Lipkin, A. Rosenfeld, pages 298–308, 1970.
- [PWR04] Wei Peng, Matthew O. Ward, and Elke A. Rundensteiner. Cluster reduction in multidimensional data visualization using dimension re-ordering. In *Proc. of the Symposium on information visualization*, pages 89–96, 2004.
- [RG93] Jun Rekimoto and Mark Green. The information cube: Using transparency in 3d information visualization. In *Proceedings of the third anual Workshop on Information Technologies and Systems*, pages 125–132, 1993.
- [RMC91] George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone trees: animated 3d visualizations of hierarchical information. In *Proceedings of the SIGCHI conference on Human factors in computing system through technology*, pages 189–194, 1991.
- [Ros56] Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. *Annals of Mathematical Statistics*, 27(3):832–837, 1956.
- [RT81] Edward M. Reingold and John S. Tilford. Tidier drawings of trees. *IEEE Transactionos on Software Engineering*, 7(2):223–238, March 1981.
- [SBG00] T. C. Sprenger, R. Brunella, and Markus H. Gross. H-blob: a hierarchical visual clustering method using implicit surfaces. *Proceedings of the conference on Visualization '00*, pages 61–68, 2000.
- [Sco92] David W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley, New York, 1992.
- [Shn92] Ben Shneiderman. Tree visualization with treemaps: A 2d space-filling approach. In *ACM Transactions on Graphics*, volume 11, pages 92–99, 1992.
- [Sil86] Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, 1986.
- [SNLH09] Mike Sips, Boris Neubert, John P. Lewis, and Pat Hanrahan. Selecting good views of high-dimensional data using class consistency. *Compute Graphics Forum*, 28(3):831–838, June 2009.
- [SR06] Harri Siirtola and Kari-Jouko Raiha. Interacting with parallel coordinates. In *Interacting with Computers*, volume 18, pages 1278–1309, 2006.

- [SS02] Jinwook Seo and Ben Shneiderman. Interactively exploring hierarchical clustering results. *IEEE Computer*, 35(7):80–86, 2002.
- [SS05] Jinwook Seo and Ben Shneiderman. A knowledge integration framework for information visualization. *Lecture Notes in Computer Science*, 3379:207–220, 2005.
- [STTX08] Yang Sun, Jiuyang Tang, Daquan Tang, and Weidong Xiao. Advanced star coordinates. In *Web-Age Information Management, 2008. WAIM 08. The Ninth International conference*, pages 165–170, 2008.
- [Stu03] Werner Stuetzle. Estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample. *Journal of Classification*, 20:25–47, 2003.
- [SY06] Jahangheer S. Shaik and Mohammed Yeasin. Visualization of high dimensional data using an automated 3d star coordinate system. *International Joint Conference on Neural Networks*, pages 1339–1346, 2006.
- [SY07] Jahangheer S. Shaik and Mohammed Yeasin. Selection of best projection from 3d star coordinate projection space using energy minimization and topology preserving mapping. *International Joint Conference on Neural Networks*, pages 2604–2609, 2007.
- [TBET99] Ioannis G. Tollis, Giuseppe Di Battista, Peter Eades, and Roberto Tamassia. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.
- [The00] Holger Theisel. Higher order parallel coordinates. In *Proceedings Vision, modeling and Visualization 2000*, pages 119–125, 2000.
- [TM02] Soon Tee Teoh and Kwan-Liu Ma. Rings: A technique for visualizing large hierarchies. In *Graph Drawing, Lecture Notes in Computer Science*, volume 2528, pages 51–73, 2002.
- [TM03] Soon Tee Toeh and Kwan-Liu Ma. Starclass: Interactive visual classification using star coordinates. In *Proc. 3rd SIAM Intl. Conf. on Data Mining*, 2003.
- [Tuf83] Edward R. Tufte. *The Visual Display of Quantitative Information (2nd edition)*. Graphics Press, Cheshire, 1983.
- [vWvL93] Jarke J. van Wijk and Robert van Liere. Hyperslice: Visualization of scalar functions of many variables. In *Proceedings of the 4th conference on Visualization*, pages 119–125, 1993.
- [WB97] Pak Chung Wong and R. Daniel Bergeron. 30 years of multidimensional multivariate visualization. In *Scientific Visualization Overviews*

- Methodologies and Techniques. IEEE Computer Society Press*, pages 3–33, 1997.
- [Weg90] Edward J. Wegman. Hyper-dimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 21:664–675, 1990.
- [WL83] Anthony Wong and Tom Lane. A kth nearest neighbor clustering procedure. *Journal of the Royal Statistical Society, Series B*, 45:362–368, 1983.
- [WMW86] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structure for soft objects. *The Visual Computer*, 2:227–234, 1986.
- [WN08] Daniel Whalen and Michael L. Norman. Competition data set and description. *IEEE Visualization Design Contest*, <http://vis.computer.org/VisWeek2008/vis/contests.html>, 2008.
- [Won82] M. Anthony Wong. A hybrid clustering method for indentifying high-density clusters. *Journal of the American Statistical Association*, 77(380):841–847, 1982.
- [Wri95] William Wright. Information animation applications in the capital markets. In *Proc. Int. Symp. on Information Visualization*,, pages 19–25, 1995.
- [YWR02] Jing Yang, Matthew O. Ward, and Elke A. Rundensteiner. Interring: An interactive tool for visually navigating and manipulating hierarchical structures. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 77– 84, 2002.
- [YWR03] Jing Yang, Matthew O. Ward, and Elke A. Rundensteiner. Visual hierarchical dimension reduction for exploration of high dimensional datasets. In *Proc. of the Symposium on data visualization*, pages 19–28, 2003.
- [ZRL96] Tian Zhang, Raghu Ramakrishman, and Miron Livny. Birch: An efficient data clustering method for very large databases. In *SIGMOD conference*, pages 103–114, 1996.
- [ZYQ⁺08] Hong Zhou, Xiaoru Yuan, Huamin Qu, Weiwei Cui, and Baoquan Chen. Visual clustering in parallel coordinates. *Computer Graphics Forum*, 27(3):1047–1054, 2008.

Declaration

I hereby declare that this dissertation was done by my own work without any impermissible assistance.

Date, Signature

