A Framework for the Visualization of Multidimensional and Multivariate Data

by

Selan Rodrigues dos Santos

Submitted in accordance with the requirements
for the degree of Doctor of Philosophy.

The University of Leeds
School of Computing

September 2004

The candidate confirms that the work submitted is his own and that the appropriate credit has been given where reference has been made to the work of others.

Abstract

High dimensionality is a major challenge for data visualization. Parameter optimization problems require an understanding of the behaviour of an objective function in an *n*-dimensional space around the optimum - this is multidimensional visualization and is a natural extension of the traditional domain of scientific visualization. Large numeric data tables with observations of many attributes require us to understand the relationship between these attributes - this is multivariate visualization and is an important aspect of information visualization.

Common to both types of 'high dimensional' visualization is a need to reduce the dimensionality for display. Although multidimensional and multivariate data are quite distinct, we show that a common approach to dimensionality reduction is possible. This framework makes a contribution to the foundation of the data visualization field, bringing both information and scientific visualization rather closer together.

To address this problem we present a uniform approach designed for both abstract and scientific data. It is based on the reduction approach, which is realized through a filtering process that allows extraction of data subject to constraints on their position or value within an *n*-dimensional window, and on choice of dimensions for display. The framework has been put to proof through a visualization method called HyperCell, which has been applied to several case studies. The results are presented and the system evaluated.

Acknowledgements

First, I thank my supervisor Professor Ken Brodlie for his guidance, advice, encouragement, and inspiration. He has been a great mentor and and an outstanding example of kindness and generosity.

I acknowledge the financial assistance of the Brazilian Ministry of Education (MEC), through the CAPES agency, who provided the scholarship (BEX 151699-0) for my PhD course.

I especially acknowledge the support given by Dr Jason Wood on several implementation issues involving IRIS Explorer, Marcelo Cohen and Ying Li who shared office with me for their pleasant company and valuable discussion on numerous topics, and Dr Chris Goodyer for his help with the Nag optimization routines. I also wish to thank the members of my examining committee, Dr. Helen Wright and Dr. David Duke, for their diligent reading of this dissertation.

I would like to acknowledge the invaluable help given by the following people, who have provided me with the real world applications for the three case studies discussed in this dissertation: Dr Vania Dimitrova and Riccardo Mazza for the CMS data; Dr Ian Coope for the helping with Rosenbrock function data and optimization algorithms; and Dr Bob Mann for the SuperCOSMOS Science Archive data. Their collaboration and patient explanation about the data and the corresponding application were crucial to the evaluation process of this work.

I would also like to thank the many friends and colleagues who have helped make life during my PhD studies in Leeds an enjoyable journey, sharing their experience and love - Thank you all!

Finally, I am deeply grateful to my fabulous family who have supported me unconditionally throughout my studies, especially my beloved wife Cati who has always been my stronger supporter, a great partner, my best friend, and a remarkable human being - *Cati, this is dedicated to you.*

Declarations

Some parts of the work presented in this thesis have been published in the following articles:-

dos Santos, S. R. and Brodlie K., "Visualizing and Investigating Multidimensional Functions", *in: D. Ebert, P. Brunet, I. Navazo (Eds.), Proceedings of the Joint Eurographics/IEEE TVCG Symposium on Data Visualization 2002*, IEEE Press/ACM Press, Barcelona, Spain, 2002, pp. 173–181, 276.

dos Santos, S. R. and Brodlie K., "Gaining Understanding of Multivariate and Multidimensional Data through Visualization", *Computers & Graphics*, 28(3) (2004) pp 311-325.

Contents

List of Figures

List of Tables

Chapter 1

Problem Description

THIS DISSERTATION AIMS to show that visualization techniques for multivariate and multidimensional data can follow a uniform framework based on a dataflow paradigm; and that these techniques can improve the understanding of high-dimensional problems through a visual interpretation of the data.

1.1 Introduction

Visualization is concerned with representing data in a graphical form that improves understanding. The major objective is to provide insight into the underlying meaning of the data. To achieve this goal visualization relies heavily on the human's ability to analyze visual *stimuli* to convey the information inherent to the data.

Traditionally visualization applications can be categorized into two broad domains: *scientific visualization* (SciVis) and *information visualization* (InfoVis). This division is primarily based on the nature of the data being visualized. Scientific visualization is primarily concerned with applications whose data originate from measurements of scientific experiments, complex simulations, or advanced mathematical models, usually with an inherent physically based component [59, 136]; whereas information visualization is mostly interested in applications whose data is defined over abstract spaces (information spaces) that frequently does not originate from intrinsically spatial data [32, 78, 79, 181, 206].

However, the precise definition for both fields and the implications of treating them separately have been subject of a recent debate, as shown in [160]. Tory and Möller in [192] have suggested a slightly different approach in defining the SciVis and InfoVis domains, aiming to reduce some degree of ambiguity that the traditional definitions may induce. For example, genetic sequence data from bioinformatics is an abstract code rather than a physical representation, therefore this application should belong to information visualization domain; but they are in fact of scientific origin, thus according to the definition they should be considered as scientific visualization applications. To avoid this double interpretation the authors suggest that the division between SciVis and InfoVis domains should be based on the characteristics of models of the data, rather than on the characteristics of the data itself, hence the suggested terminology: *continuous model visualization* – encompassing visualizations that deal mostly with continuous data model (thus roughly associated with SciVis); and, *discrete model visualization* – encompassing visualizations that deal mostly with a discrete data model (thus roughly associated with InfoVis).

Of particular interest for this research are applications involving high-dimensional data that occur in both visualization domains and the primary goal of this work is to investigate visualization methods for such a category of data. A high-dimensional visualization method must overcome the difficult problem of converting the complex high-dimensional data into an appropriate low-dimensional representation for display.

1.1.1 Defining Terminology

It is important to define our terminology, since words such as 'dimensionality', 'multidimensional', and 'multivariate' are overused in the visualization literature.

The term *variable* is at the core of these definitions. An item of data is composed of variables, and when such a data item is defined by more than one variable it is then called a *multi*variable data item. Variables are frequently classified into two categories: *dependent* or *independent*[1]. The exact definition for the terms dependent and independent variables varies slightly among mathematicians, engineers, statisticians, social scientists, etc. Nonetheless, it is possible to distinguish two major definitions for dependent/independent variables, explained next.

For physicists and statisticians a variable is a physical property of a subject, such as mass, length, time, etc., whose quantity can be measured. If a dataset is composed of variables that follow this definition our goal is to understand the relationships among the multiple variables. Commonly a dataset of this sort originates from an experiment in

[1] Statisticians use the corresponding terms *response* variables and *predictor* variables, respectively (see, for example, [131, page 233]).

which the independent variables are manipulated by the experimenter and the dependent variables are measured from the subjects. Sometimes these terms take a somewhat different interpretation in the sense that they are applied in studies where there is no direct manipulation of independent variables. Rather the study assigns subjects to "experimental groups" based on some pre-existing properties of the subjects. For instance, a statistical experiment may require one to ascertain if males are more inclined to car accidents than females. In this case the gender could be called the independent variable, whereas the statistical data regarding accidents would be considered as the dependent variables.

For mathematicians, however, variable is usually associated with the idea of *physical space* – often an n-dimensional Euclidean space $\mathbb{R}^n$ – in which an 'entity' (for example a function) or 'phenomenon' of continuous nature is defined. Data location within this space may be referenced through the use of a range of coordinate systems (e.g. Cartesian, polar). The dependent variables are those used to describe the 'entity' (for example the function value) while the independent variables are those that represent the coordinate system used to describe the space in which the 'entity' is defined. If a dataset is composed of variables whose interpretation fits this definition our goal is to understand how the 'entity' is defined within the n-dimensional Euclidean space $\mathbb{R}^n$.

Sometimes we may distinguish between variables meaning measurement of property, from variables meaning a coordinate system, by referring to the former as ***variate***, and referring to the latter as ***dimension***. Hence the definitions below:-

Definition 1.1 (Multi*variate* dataset) *is a dataset that has many dependent variables and they might be correlated to each other to varying degrees. Usually this type of dataset is associated with* discrete data models.

Definition 1.2 (Multi*dimensional* dataset) *is a dataset that have many independent variables clearly identified, and one or more dependent variables associated to them. Usually this type of dataset is associated with* continuous data models.

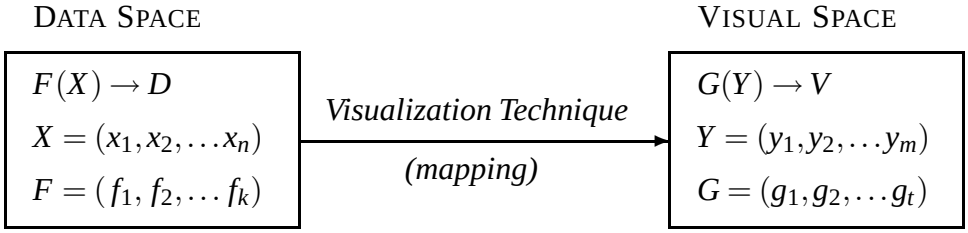
Some of the elements of our terminology, such as *multivariate* and *multidimensional*, have been partly based on the definitions introduced by Wong and Bergeron in [212] and influenced by the terminology introduced by Tory and Möller in [192]. Another observation regarding terminology is that whenever we refer to a dataset in a more general way, regardless of its origin or interpretation, we shall use the term *multivariable* dataset.

Definition 1.3 (Multivariable dataset) *is a dataset that can be either* multivariate *or* multidimensional. *This dataset is particularly called* high-dimensional *dataset if the dataset contains more than three variables (dependent and/or independent).*

Formally, we shall think of an item of data as a sample from a k-variate function $F(X)$ defined over an n-dimensional domain D – this is the *data space*. Thus $F = (f_1, f_2, \ldots f_k)$ has k components, and $X = (x_1, x_2, \ldots, x_n)$ is a point in D. We shall allow k to be zero, in which case we just have a point in D, and we allow n to be zero in which case we just have a value of F. We shall talk in terms of *dependent* variables F and *independent* variables X.

1.1.2 The Visualization Process

Ultimately any high-dimensional visualization technique maps the *data space* – regardless of being either multivariate or multidimensional – into a multivariable *visual space*. The distinction between the various visualization techniques resides in the way they create such mapping[2].



The ***visual space*** is defined in a similar manner to the ***data space***. We shall think of an item of *mapped* data as a sample from a t-variate function $G(Y)$ defined over an m-dimensional domain V – this is the visual space. Thus $G = (g_1, g_2, \ldots, g_t)$ has t components, and $Y = (y_1, y_2, \ldots y_m)$ is a ***visual mark*** in V. We shall allow t to be zero, in which case we have just the visual mark without any ***graphical property*** (such as colour, size, brightness, etc.) in the geometric space V, and we allow m to be zero in which case we just map the original data to graphical properties without using the ***spatial location*** of the visual mark to encode information (e.g. an array of *Chernoff faces* [41]). We shall talk in terms of *dependent* variables G meaning the graphical properties of the visual mark, and *independent* variables Y meaning the spatial location of the visual mark.

To recap, the *visual space* created by a visualization technique is composed of three elements: a coordinate system representing spatial location (i.e. the three-dimensional space we can represent on a display); a visual mark (i.e. an object located in the three-dimensional *visual space*); and, the graphical properties of a visual mark. The value m of the mapping determines the number of spatial coordinates used to locate a visual mark in

[2] Card *et al.* in [34] also describe the visualization process as a mapping of *data tables* into *visual spaces*.

the created *visual space*; and the value t determines the number of graphical properties of the visual mark – e.g. the size of the visual mark (0-D - meaning points in space; 1D - lines; 2D - areas; and, 3D - volumes), colour, texture, orientation, brightness, etc.

Consider the following multivariate and multidimensional examples, respectively:-

- A two-*variate* dataset, with two observations: $O_1 = (\alpha_1, \beta_1)$, and $O_2 = (\alpha_2, \beta_2)$; has $k = 2$ and $n = 0$. If a scatterplot representation is used to visualize this dataset – mapping α to the X Cartesian coordinate, and β to Y Cartesian coordinate – the final result are two points on the display. Therefore in the visual display we have $t = 0$ (no dependent variables), and $m = 2$ (the two-dimensional coordinate dimensions used to locate the two points on the display).

- A two-*dimensional* dataset corresponding to temperature measured on a regular grid over a terrain has $k = 1$, the temperature, and $n = 2$, the coordinates of grid over the terrain. If a height-field, represented by a coloured smooth surface, is used to visualize this dataset we have $t = 2$, meaning the two graphical properties of the surface (height and colour) that have been used; and $m = 2$, meaning the X and Y axes used to locate the sampled data within the *visual space*. Such surface can be generated via interpolation only because the model of the data is known to be continuous throughout the terrain, since the temperature exists all over the area.

1.1.3 The Application Domain of Scientific Visualization

Scientific visualization commonly deals with multidimensional visualization. Usually the visualization is concerned with sample data which is given at specified points within the *n*-dimensional domain *D*, and the goal is to recreate from this sampled data an estimate of the underlying entity, $F(X)$, over the entire domain. Interpolation is a key part of this process. In mathematical modelling applications, the model itself may be provided to us, as an approximation to some physical phenomenon that is being investigated. Corresponding datasets may be generated during a pre-processing step by evaluating the model at a set of points in the *n*-dimensional domain *D*.

Often the number of dimensions is small – from simple 1D applications such as temperature measured at different times, to 3D applications such as medical imaging, where data is captured within a volume. Standard techniques – contouring in 2D; isosurfacing and volume rendering in 3D – have emerged over the years to handle this sort of data [25, 167, 172]. For these applications the data and display dimensions essentially match, thus there is no problem in finding a suitable visual representation.

Increasingly, however, scientific visualization needs to concern itself with higher dimensionality problems, such as occur in parameter optimization problems, where we wish to visualize the value of an objective function, $F = (f_1)$ in terms of a large number n of control parameters, $X = (x_1, x_2, \ldots x_n)$, say. This is a much harder problem and relatively unexplored.

It is also not uncommon for scientific visualization to deal with multivariate numerical data, as for example the case in which we are interested in performing a chemical analysis of an ice core extracted from the Antarctic ice sheet looking for signs of previous global warming. In this case $n = 0$, because the data is gathered from a single object whose spatial location does not affect the final outcome (since they are interested in an historical view of the ice core, which is independent of the location where the ice core was taken), and the measurements become a set of k-tuples (where k is the number of chemical parameters being observed) with cardinality (i.e. the number of observations) S.

1.1.4 The Application Domain of Information Visualization

Information visualization commonly deals with multivariate data from application areas such as statistical analysis, stock markets, census data, etc. In many applications, the data is given in the form of a data table, where each column represents an attribute, and each row represents an observation of these attributes. The number of variates is typically quite large and the attributes may be categorical or numerical. There are no independent variables here (in the sense of space coordinates), so we can view n as zero, and see the data as an unordered set of k-tuples with S elements, $F^i = (f_1^i, f_2^i, \ldots f_k^i), i = 1, 2, \ldots, S$. In fact it is possible to make an alternative, geometric interpretation, and think of these as S points in a k-dimensional space – this contributes to the ambiguity of the term 'dimension' in visualization. Another observation on terminology is pertinent here: sometimes we may refer to the components f_k^i as *attributes*, in addition to the term variate; and a data item sometimes may be called an *observation*.

The goal of multivariate visualization depends on the context of the problem but it usually involves the searching for patterns, structure (clusters), trends, behaviour, or correlation among attributes [13, 97, 98, 209]. The resulting information is then fed into the exploratory stage of the knowledge-acquiring process to support the elaboration of a hypothesis about the phenomenon responsible for the targeted data. Hence visualization has been considered a helpful tool in augmenting analytic approaches of multivariate data, especially during the exploratory stages of the data analysis process.

1.1.5 Multidimensional and Multivariate Data

There are further applications which are both multidimensional and multivariate, i.e. according to our definition they are multidimensional but they also have elements that characterizes a multivariate data. For example, in medical imaging we may wish to look at co-registered CT and MR data: here we have two variates defined over a 3D domain [194]. The numbers of variates and dimensions are small in this case, and so it is possible to solve this particular application by extension of existing methods, for example, combining the two variates in some way within the volume rendering process.

Other application is in optimization where in addition to many parameters, there may be several different criteria – thus again, several variates and several dimensions. For a discussion of optimization problems with multi-criteria objective functions, see for example [153].

Another example from optimization is the visualization of trajectories of intermediate estimates of the solution point, as generated by an iterative algorithm. The algorithm generates a sequence of points $\{Y^i\}, i = 1,2,\ldots S$ towards the *minimum* of a function $Q(Y)$, of k variables $y_1, y_2, \ldots y_k$. The points can be regarded as S items of multivariate data with k attributes, but they are ordered in sequence. However we also know the value of the objective function Q at each point: visualization of the function is a multidimensional visualization problem.

1.1.6 Multidimensional vs. Multivariate Visualization Approaches

A frequent approach used to visualize multivariate data is to choose some variates from the *data space* and map them onto a coordinate system in the *visual space* (i.e. the independent variables m of the *visual space* mapping function G). Because the variables of multivariate data usually are measurements of certain properties and not spatial dimensions, such mapping might result in visual marks being located only within a limited region of the *visual space*. This is, actually, an effect of a phenomenon called the *curse of dimensionality* [18] that refers to the exponential growth of the high-dimensional space (i.e the 'hypervolume') as a function of dimensionality. Consequently multivariate spaces of increasing dimensionality generated by finite samples tend to be sparse.

In contrast, multidimensional data is normally obtained from an entity that is defined continuously over the n-dimensional domain D. Thus mapping the independent variables n of the *data space* onto independent variables m of the *visual space* is considered a sensible approach, apart from the situation when $n > m$ in which case a more elaborate solution is required.

1.2 Motivation

The problem of finding an effective representation of a high-dimensional entity is a hard problem to solve. As the dimensionality of a high-dimensional dataset increases, so does the complexity of finding an effective visual representation that promotes insight into the underlying meaning of the data.

The difficulty originates from the difference between the dimensionality of a high-dimensional *data space* and the maximum number of independent variables *m* of the *visual space*. Thus the perceptual advantage of using the spatial representation as the primary choice for mapping variables from the *data space* is limited by a maximum number of, say, three spatial dimensions of the display. The remaining variables of the *data space* have to be mapped to other dependent variables of the *visual space* such as shape, colour, orientation, texture, etc. (hence the increasing importance of perceptual issues in visualization).

In the last decades there have been several proposals of novel visualization approaches to deal efficiently with this problem of finding a suitable mapping [37,109,212]. Each one of them has addressed the problem using different strategies, but, according to Hibbard [93], "each technique reveals some information but no technique reveals all information."

These factors, the challenge of finding a suitable mapping for high-dimensional data and the increasing production of high-dimensional data have been the general motivations behind this research, whose objective is to study the area of high-dimensional visualization, looking for a more comprehensive understanding of how the visualization process works to reveal insight.

1.3 Research Problem & Scope

The research was initially driven by the particular problem of visualizing scalar functions of several variables. Subsequently the research was extended to encompass also the problem of visualizing numeric multivariate data.

At a later stage, however, we realized that even though the structures of both multidimensional and multivariate data spaces may differ, they underwent similar steps during the process of reducing its dimensionality for display.

Consequently, we recognized that it would be possible to elaborate a general framework to accommodate both types of data if we ignored the inherent structural distinction between multivariate and multidimensional data spaces and focused on the similar core visualization processes.

> *The main **research problem** is to design, develop and evaluate a visualization method capable of providing insight into a high-dimensional numeric data set, based on a general framework that incorporates both abstract and scientific numerical data.*

The focus of this research is mainly on numerical data, but the results can be extended to categorical data whenever is possible to apply a mapping of the categorical range onto an integer numerical. Also we have chosen to tackle problems whose number of dimensions are not excessively high, having a upper limit of approximately 60 dimensions. Figure 1.1 shows how the scope of this work relates to both scientific and information visualization. Note that multivariate numerical data occur in both scientific and information visualization fields.

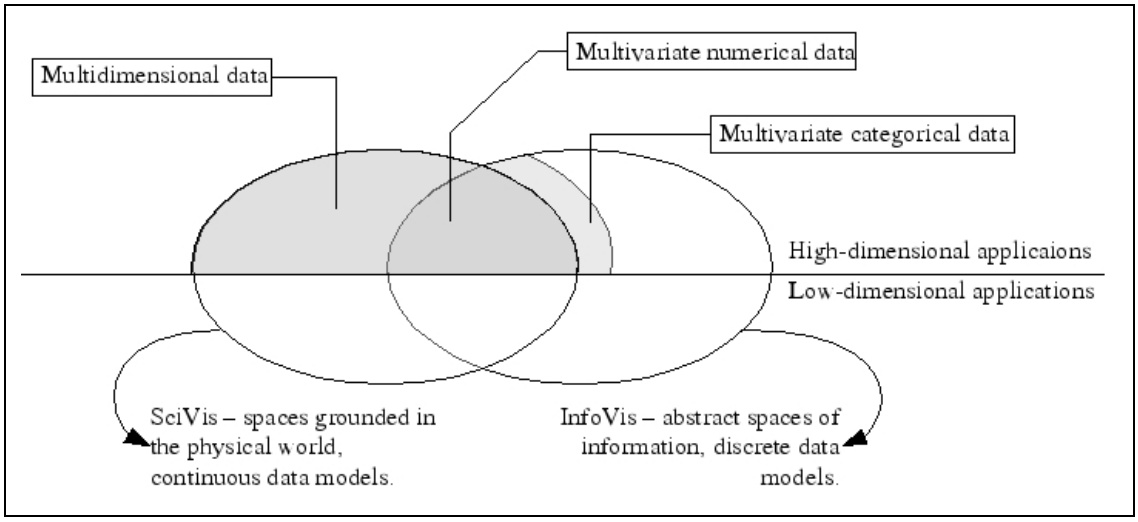


Figure 1.1: This diagram shows, in grey, the data category we are interested in and how it relates to both scientific (SciVis) and information visualization (InfoVis) fields.

1.4 Methodology & Proposed Solution

We have followed a typical methodology for applied sciences [202]. The first phase is concerned with the identification of the problem, which triggers the second phase: a research into the related disciplines. The aim is to identify existing theories, methods and models that could be adapted and applied to the research problem. In the third phase the resources found are extended and adapted in such a way as to be useful within the field of study. Next, an initial theory is constructed, an abstraction is made towards a common framework that supports the proposed solution, usually as a result of the integration

or adaptation of prior theories. The implementation of the proposed solution and its application to existing problems comes next. Following that the results are collected and evaluated; the new theory is then revisited, augmented and tuned, which prepares it to be used as a starting point for further research in the field.

The subject of this thesis is an investigation into methods for representing graphically high-dimensional data and how these methods relate to existing visualization models. The investigation has been based on resources gathered and adapted from prior relevant work in scientific visualization, information visualization, and human computer interaction. Therefore many of the concepts are not new but their extension and presentation under the new framework are novel.

The basic strategy of our proposed visualization method is the *reduction* approach [37, 40, 115]. This approach basically tries to reduce the dimensionality of data presented from n to two or three dimensions, which then can be easily mapped onto flat displays. The reduction is done by presenting several subspaces that can be generated either considering only a subset of the original dataset (thus temporarily ignoring the rest of it); or by means of some mathematical treatment, thus creating a new dataset derived from the initial source.

We have chosen the reduction strategy for the following reasons: 1) no method has ever succeeded in representing a multidimensional dataset in full resolution using a single view; 2) it seems reasonable to adopt a strategy that has already been used in both multivariate and multidimensional types of data (e.g. *scatterplot matrix [39]* in the multivariate realm, and *hyperslice* [200] in the multidimensional realm); and finally, 3) choosing a strategy that works for both types of data supports a better level of integration in our proposed framework.

The overall results from the application of our method to high-dimensional problems have been collected and organized in such a way as to form the basis for the assessment of this work.

1.5 Goals

The overall goals of this research are:-

- Review the field of high-dimensional visualization in both scientific and information visualization domains to identify and classify the underlying strategies that have been used to solve the research problem.

- Investigate a new reference model that accommodates both multivariate and multidimensional data types.
- Design and develop a new visualization technique that addresses the problem of visualizing high-dimensional datasets.
- Evaluate our proposed visualization method through application to a range of real-world problems.
- Bring the two fields of scientific and information visualization rather closer together, by providing a common framework for multivariate and multidimensional numeric data.

1.6 Overview of Dissertation Contents

This dissertation has the following structure:-

Chapter 2 (Field Review) presents an up-to-date overview of how visualization methods have been used to tackle multivariate and multidimensional data. The chapter reviews some classification proposals that have been put forward as a means of helping to identify the similarities and common features behind the design of high-dimensional visualization techniques. A new classification scheme is then suggested and some visualization techniques are classified using the suggested scheme. The last part of the chapter provides a review of relevant visualization methods.

Chapter 3 (Analysis of the Problem) puts this work into perspective by exploring in more detail the issues related to the main research question. It delves into conceptual issues involved in perception, insight, interaction, and multiple views. The Haber-McNabb dataflow reference model [88] originally designed to help understand scientific visualization applications is revisited and expanded to incorporate the treatment of high-dimensional data. Based on the suggested reference model, we identify a key filtering process that has been designed to reduce the complexity of the problem.

Chapter 4 (Description of the Proposed Solution: part 1) describes how the framework presented in the preceding chapter is realized. This chapter focuses only on the first part of the proposed visualization technique which describes the definition of the filtering parameters and the extraction of subspaces of the data. Each process of the filtering process is described in detail and some examples are given to illustrate its functionality. The chapter also sets the scenario for the next chapter which covers the remaining elements of the proposed visualization technique.

Chapter 5 (Description of the Proposed Solution: part 2) continues the description of the visualization technique initiated in the previous chapter. It addresses the issues of building the user's mental model of the complex data based on various subspaces with reduced dimensionality, as a result of the filtering process. Another set of issues are identified such as navigation, continuity in *n*-dimensional space, and layout of subspaces. The corresponding modules are described and again examples are given to illustrate their functionality.

Chapter 6 (Description of the Proposed Solution: part 3) finishes the description of the proposed solution by delving into technical issues such as implementation environment, and the schematic organization of the visualization system. This chapter also lists the advantages and limitations of the current implementation.

Chapter 7 (Advantages and Disadvantages of the Proposed Solution) starts by providing some background information on the evaluation of visualization systems. Then it presents the methodology adopted for this work which basically involves two stages: *design evaluation*, and *usability inspection* performed in three case studies involving real world applications: understanding the behaviour of scalar function of many variables and the trajectory of solutions from optimization algorithms, calibrating an astronomical catalogue, and supporting the development of distance education courses through Course Management Systems (CMS) data. The chapter finishes with the gathering of the results obtained during the evaluation process.

Chapter 8 (Conclusion and Future Work) summarizes the work undertaken, revisiting the goals of this research, how they have been achieved, identifying the principal lessons learned, and stating the contributions made by this work. This chapter ends with a list of topics to be tackled in future work.

Chapter 2

Field Review

GAINING INSIGHT ABOUT data is not just a matter of 'presenting' it, but 'translating' *data* into *information*, as observed by Spence [181]. This 'translation' should enable those engaged in the data visualization process to either form, enhance, and/or modify a cognitive model of the entity or phenomenon represented by the data. But before the formation of such a cognitive model takes place one needs to perceive the data.

Data visualization is a mental activity whose aim is to aid this perceptual process through vision, the human sense with greatest information bandwidth [206]. The main concept is to take advantage of the human perceptual system in 'pre-processing' the information before the cognitive system takes over control. In fact data visualization has been around for almost a millennium, aiding the investigation of complex problems[1].

The majority of the research in both *scientific visualization* (SciVis) and *information visualization* (InfoVis) is concerned with the visualization of low-dimensional data[2], as reported in [167, 172] for SciVis applications, and in [181] for InfoVis applications. Nonetheless there is a growing need for the visualization of data sets with dimensionality greater than three in both fields. Consider for instance the three scenarios below:-

- Parameter optimization problems [153], in which one needs to understand the behaviour of an objective multidimensional function within a specific region in the

[1] See for example the work of Tufte [196] and Collins [54] for examples of early graphical models on paper and their impact on various areas of the human activity.

[2] By low-dimensional data we mean one-, two-, or three-dimensions.

n-dimensional space – for example, in the vicinity of a *minimum* point.

- The problem of *design steering*, in which one needs to understand the *n*-dimensional parameter space created by several design parameters of the product being manufactured in relation to the performance measurements of that product. The main goal here is to use visualization to support the optimization of those parameters by relating them to the product design – this is an interesting application because one may contemplate it as a multidimensional application in which we have *n* independent design parameters that yield *k* performance measurements; or regard it as a multivariate application by generating *S* tuples corresponding to several design trials and the corresponding results, in which case each tuple would have the values for *n* parameters together with the *k* performance measurements. For further information on this type of application we refer the reader to the results obtained by Tweedie *et al.* [198] and the work done in the DIVA project [214].

- The problem of visualizing data that can be organized in a 'table' format, i.e, each column represents an attribute reflecting some sort of property measurement (this could be simple categorical data, or real-valued data), and each row contains an observation of these attributes. For instance, the data generated by *course management systems* (CMS) – software used to support distance learning courses – usually are the basis to evaluate the overall performance of a group of students. This type of data normally contains information such as individual student performance, attendance frequency, number of access to the topics of the course, number of virtual group meetings, number of messages exchanged in the course so far, etc. [135]

The first scenario may well be regarded as typical SciVis application (scientific origin and continuous model of data); the second scenario can be treated either as a SciVis (continuous model of data) or a InfoVis (when data comes from abstract origin) application; and the last one characterizes an InfoVis application (abstract origin and discrete model of data). All three scenarios describe important and increasingly common applications for which there is no obvious visual mapping of their high-dimensional entities/spaces to the three dimensions conventionally emulated on a two-dimensional display.

Most of the research done on the design of visualization methods until mid 1990s concentrated mostly in improving the algorithms that map data into images often neglecting the importance of the perception principles [83, 159]. They also lacked both a solid basis in their design and a systematic pursuing of any empirical validation [82]. Consequently this situation has increased the need for a more formal base for the visualization field, which is an old quest for the field, as suggested by DeFanti *et al.* [59] and Rosenblum *et*

al. [167, Chapter VII]. The proposal of classification schemes, and the design of models that describe the visualization process in its various levels of abstraction are key factors that contribute to form a foundation for the visualization practice.

In this chapter we firstly review early efforts towards a classification scheme for high-dimensional visualization methods, highlighting their virtues and weaknesses. Then we introduce our own classification scheme in an attempt to overcome most of the shortcomings of the schemes described previously. In the last part of this chapter we review in more detail some visualization techniques that are relevant to this work.

2.1 High-dimension Visualization Classification

A classification mechanism based upon a well defined set of categories is a useful tool for several reasons, (a) it improves the understanding of how proposed techniques relate to each other, by looking at similarities and dissimilarities of techniques in the same category; (b) it clarifies the driving ideas and objectives behind the design of classified techniques, because they have to be understood to warrant a correct classification into categories; (c) it may encourage the augmenting of existing techniques, by comparing less successful techniques with successful ones within the same category; and finally, (d) it organizes seemingly unrelated techniques which may even stimulate the devising of new methods to deal with situations for which there are no existing solution [29, 33].

Classification schemes for visualization methods may follow one of several different strategies or combine them into a hybrid approach. We have identified four different strategies:-

- ***Entity-based:*** This classification approach draws its categories from the type of data representing the entity that we wish to visualize. Besides the data type an entity is defined by an empirical model associated with it. Once categories of entities are established, techniques may be associated with them. This is usually carried out based on an empirical assessment of a technique's performance in conveying information for entities in a given category.

- ***Display-based:*** This type of classification defines the categories from the features in the visual display of a method that can be used to differentiate one method from another. Examples of such features are the dimensionality of the visual representation, or the appropriate use of perception issues.

- ***Goal-based:*** A goal-based classification establishes its categories according to the purpose of the visualization, i.e. the objective we wish to achieve by using vi-

sualization as a tool to convey information. This type of classification is notably suitable for helping the domain specialists in selecting visualization methods that are pertinent to accomplish the objectives listed by the classification. An example of such classification in the SciVis field is presented by Keller and Keller [111], which provides several examples of visualization goals in the context of practical applications.

- ***Process-based:*** In this approach a classification designates its categories according to the sequence of procedures or steps done during the whole process of visualization. Here the action required to generate a visualization is the focus.

Next we review a few examples of classification schemes representative of each of the four strategies just described.

2.2 Early Classification Proposals

The classification schemes we reviewed, organized by strategy, are the following:-

1. Entity-based: *E* notation, introduced by Brodlie [25, Chapter 3] and [23].
2. Display-based: Wong and Bergeron classification [212].
3. Goal-based: Task by Data Type Taxonomy (TTT) by Shneiderman [176].
4. Process-based: Data visualization taxonomy by Buja *et al.* [29], and the classification introduced by Keim [108, 109].

The criterion used to select the classification schemes reviewed in this section was their relevance for the field of high-dimensional visualization based on the number of citing references in the literature[3].

The selected classification schemes are presented next, in a chronological order. But before proceeding with this review we have to make an observation regarding referencing. During the review of those classification methods several techniques are repeatedly mentioned by the authors. To avoid the inconvenience of making the reader move back and forth to the Bibliography chapter every time a technique is mentioned we grouped the techniques and their corresponding references in Table 2.8 located at the end of this chapter.

[3]Buja *et al.* - 38 citations; Wong and Bergeron - 9 citations; and, Shneiderman - 53 citations, *source*: www.citeseer.com. Brodlie *et al.* - 5 citations; and Keim - 6 citations, *source*: portal.acm.org.

2.2.1 *E* Notation by Brodlie

In 1992 Brodlie presented a classification scheme for scientific visualization (sometimes called the *E* notation). The basic concept was that a visualization is concerned with the entity we need to visualize rather than the data alone. An entity is represented by the acquired data and an empirical model associated with the entity.

The classification therefore is based on two elements: the type of data representing the entity, and the dimension n of the entity's domain. There are four categories based on the entity's data type: scalar (denoted by the letter S), set of points (P), vector (V_m, where m is the vector's length), or tensor ($T_{m:m:\ldots m}$, where the quantity of ms represents the tensor's order and the value of m is the dimensionality of the tensor). These letters are used as superscripts of an entity represented by the letter E.

The next classification is based on the dimensionality of the domain, indicated by an integer number used as subscripts of an entity E. This subscript may receive further classification according to the nature of an entity. It can be defined: (a) point-wise over a continuous domain, represented by n, where n is the entity's dimensionality; (b) over regions of a continuous domain, which is represented by $[n]$; or, (c) defined over an enumerated set, which is represented in the form $\{n\}$.

Table 2.1 shows the description of typical entities with their corresponding E notation, followed by a suitable visualization method. This should clarify the use of the E notation.

2.2.1.1 Critique

The major advantage of this classification scheme is that it charts the underlying field, making it possible to produce a comprehensive mapping of visualization techniques onto entities. This would promptly identify entities for which there is no associated technique.

This notation may be used in a straightforward manner, to determine the entities of interest for this study: if the dimensionality of the superscript component is higher than one it indicates multivariate data, whereas if the dimensionality of the subscript component is higher than one it signifies multidimensional data. For example, multivariate datasets can be of the following types: E_1^{mS}, which means multiple scalar values defined over a continuous one-dimensional scalar domain (this is the case, for instance, of chemical measurements taken along an ice core); $E_{\{1\}}^{mS}$, which means multiple scalar values defined over categorical data (e.g. collection of several attributes measured for a list of car types); and, $E_{[1]}^{mS}$, which means multiple scalar values defined over ranges on a one-dimensional domain (e.g. census data organized by age ranges). Multidimensional scalar data, on

Entity	*E Notation*	*Visualization method*
1D, 2D, or 3D multivariate data	E_1^P, E_2^P, E_3^P	Scatterplots
*n*D multivariate data	E_n^P	Andrews curves, Chernoff faces
A set of points sampled over a continuous 1D scalar domain	E_1^S	Line graph
A set of points sampled over *m* different continuous 1D scalar domain	E_1^{mS}	Multiple line graph
List of values associated with items in a enumerated set	$E_{\{1\}}^S$	Bar chart; pie chart
List of values that can be associated to ranges defi ned over a continuous 1D scalar domain	$E_{[1]}^S$	Histogram
Set of values sampled over a continuous 2D scalar domain	E_2^S	Line base contouring; discrete shaded contouring; image display; surface view
Two separate sets of values sampled over the same continuous 2D scalar domain	E_2^{2S}	Coloured height-fi eld plot
Multiple sets of values sampled over the same continuous 2D scalar domain	E_2^{mS}	Stick fi gure
Scalar fi eld sampled over a continuous 3D scalar domain	E_3^S	Volume rendering; isosurfacing
2D vector fi elds	$E_2^{V_2}$	2D arrow plots; 2D streamlines and particle tracks; 2D vector fi eld topology plot
3D vector fi elds over a 2D plane	$E_2^{V_3}$	3D arrows in plane
3D vector fi elds in a volume	$E_3^{V_3}$	3D arrow in a volume; 3D streamlines and particle tracks
Second order tensor in 3D	$E_3^{T_{3:3}}$	Symmetric, second order tensor display

Table 2.1: Listing examples of entities with their corresponding E notation and visualization techniques associated for each category according to Brodlie.

the other hand, are represented by E_n^S, and multivariate multidimensional scalar data is represented by E_n^{mS}.

Because this notation was originally designed to classify the underlying field we want to investigate, it does little to benefit the understanding of the several strategies behind high-dimensional visualization methods used to provide the visual representation for such fields.

2.2.2 Data Visualization Taxonomy by Buja *et al.*

In 1996 Buja *et al.* introduced a taxonomy for data visualization that, at a high level, is divided into two categories[4]: *rendering* and *manipulation*. This taxonomy is regarded as a process-based type of approach because the main categories are the processes involved in generating a visualization.

The taxonomy is more focused on classifying approaches or aspects of a visualization technique, rather than the technique as a whole. This means that a technique may fit into several subtypes in each category of the taxonomy.

The *rendering* category describes the features of a static image and it is divided into three subtypes as follow:

1. **Scatterplots**: the observations are mapped to location of points in two- or three-dimensional spaces;

2. **Traces**: the observations are mapped to functions of a real parameter. Examples of this type of display approach are the *parallel coordinates* and *Andrews curves.*

3. **Glyphs**: the observations are mapped to complex symbols whose features represent attributes of the observations. Examples of this visual representation are *trees and castles, Chernoff faces, shape coding*, and *stars*. Then a decision has to be made with respect to the lay-out of the glyphs. It may be appropriate to associate the spatial location of the glyphs with one of the data dimensions – possibly with the independent dimensions, if such exist. The positioning of glyphs on the display is an important step because it supports the comparison of the glyphs generated from the mapping step.

The authors emphasize the importance of the interaction aspect of the visualization to accomplish a meaningful data exploration. The idea of using visualization enhanced with interactive features to explore the data and gain knowledge is a methodology called Exploratory Data Analysis (EDA) introduced by Tukey [197] and then further explored by Cleveland and McGill [49]. The work by these authors has had a major influence on subsequent methods for visualization as an aid to statistical analysis. Therefore Buja *et al.*'s taxonomy is more detailed and centered around the *manipulation* category, which is organized in terms of three basic search tasks that it should support: *finding Gestalt*, *posing queries*, and *making comparisons*. They also proposed a correlation between these

[4]We have adopted the term *category* to keep consistency of terminology, but the original term used by the authors in their work was *area*.

three tasks with typical methods found in EDA, respectively named *focusing*, *linking*, and *arranging views*.

2.2.2.1 Finding Gestalt

This typically involves the search for some structure in the data or relationship between attributes. Examples of such a task are: find local or global linearities and nonlinearities; identify discontinuities; and, locate clusters, outliers, and unusual groups.

The authors then associate this task with a category of tools, namely *focusing individual views*. They compare the functionality of this category of tools to the action of setting up a camera and deciding which view to look at. Put differently, this is the stage in which the variables (or the projections) for viewing are chosen; or the aspect ratio, and zoom and pan parameters are set up. All this is accomplished usually in an interactive fashion.

Because these parameters sometimes can be of continuous nature (e.g. choice of projection, zoom and pan parameters) they can be animated smoothly. Examples of the application of animation of Gestalt parameters are (a) the *grand tour*, a technique that presents a dynamic sequence of projections (normally visualized as scatterplots) of the data onto a low (≤ 2) dimensional plane moved along a continuous path in the *n*-dimensional variate space; (b) the *projection pursuit* method, which is an exploratory data analysis tool that tries to find interesting low-dimensional projections of multivariate data (i.e. clusters) by optimizing a projection index (a specific function associated with the method); and, the *Exvis* project, which permitted the animation of the *stick figure*'s[5] parameters to find visually interesting textures.

2.2.2.2 Posing queries

This task tries to make sense out of the findings (i.e. views) from the *finding Gestalt* stage, usually via a graphical query posed on these views. They argue that *linking multiple views* is a representative approach for this task, which is illustrated by the *brushing*[6] technique first used in the *M and N plot*, to become almost a standard interaction tool for several visualization methods. In this technique one uses one of the views of the data to select elements (query formation); immediately the corresponding elements are highlighted on the other existing views, yielding a graphical response to the 'query' posed.

[5] A stick figure is a *m* limbed icon having each limb feature (such as length, thickness, colour, and angle) mapped to a data observation with up to *m* variables. The whole collection of icons put side by side generates a texture in a complex image, which relies on human ability of recognizing patterns to find interesting features in the data [156].

[6] Also known as *painting* [137].

The use of *brushing* in linked views is a powerful device because it may help to identify correlations between dimensions of a dataset. Consider, for example, a case in which a linear behaviour between two variables, say X and Y, is observed in one view – the 'query' view. Highlighting this linear behaviour in the 'query' view and observing the corresponding result in the other views might bring out a similar linear behaviour involving other dimensions of the original set, thereby expanding the original correlation between variables X and Y.

2.2.2.3 Making comparisons

This task involves the comparison of several views of the data (i.e. related plots of data) generated in the *finding Gestalt* task. The goal is to facilitate meaningful comparisons, and the authors call this process *arranging views*.

The several views generated during this task require some organization strategy to facilitate understanding. For views with two variables the most common arrangement is the matrix-like organization that combines the variables in pairs (e.g. *scatterplot matrix*).

We have identified other options for views with more than two variables, thus expanding the original work done by the authors. These options are nesting variables within variables (e.g. *worlds within worlds* and *hierarchical axis*), organizing the views in a spreadsheet-like format (e.g. *table lens* and *spreadsheet-like interface for visualization exploration*), applying distortion to accommodate the detail and overview (e.g. *fisheye views* and *the perspective wall*), or make use of dynamic organization (*rapid serial visual presentation - RSVP*[7]).

2.2.2.4 Applying the taxonomy to techniques

The authors did not provide full examples of how the taxonomy should be applied to the mentioned techniques, restricting themselves to classify only three techniques according to the *finding Gestalt* sub-category of the *manipulation* category.

Below we provide Table 2.2 that summarizes their textual description of the examples. The elements under the *posing queries* and *making comparisons* columns are our interpretations for those techniques regarding these categories.

[7] This technique presents successive views in a brief period of time to support the browsing of views allowing a quick comparison.

Technique	**Finding Gestalt** *Focusing individual views*	**Posing queries** *Linking multiple views*	**Making comparisons** *Arranging views*
Scatterplots (scatterplots sub-category)	Choice of projections, aspect ratio, zoom and pan	Brushing	Matrix-like arrangement
Parallel coordinates and Andrews curves (traces sub-category)	Choice of variables, their order, their scale, and the scale and aspect ratio of the plot	Brushing, hierarchical brushing	Single view
Glyphs (glyphs sub-category)	Choice of variables and their mapping to glyph features	Brushing	Comparison of glyphs with distinct mappings

Table 2.2: Listing some techniques classified according to the Buja *et al.* taxonomy. Note that the emphasis is given to the *manipulation* category. The classification under the *rendering* category is provided in brackets after the technique's name.

2.2.2.5 Critique

The Buja *et al.* taxonomy is innovative in the sense that it contains three high level sub-categories for the *manipulation* category. This introduces a degree of abstraction and allows the comparison of different techniques according to their capabilities in dealing with the three manipulation processes: focusing individual views, linking multiple views, and arranging views. Note that the use of processes as categories in this taxonomy indicates a feature of a process-based type of classification.

Another positive aspect is the use of the Gestalt theory (which has been studied for over 80 years) as a formal basis to describe the interaction (*manipulation*) part of a visualization. This contributes positively to the visualization field in the same way that the perception theory has influenced the visual design of methods in the field, providing guidelines, frameworks, and models.

However, their taxonomy presents some fundamental deficiencies. Firstly, the *rendering* sub-categories are unable to describe some valuable visualization methods, such as *dimension stacking* and *pixel-oriented methods.* Secondly, the sub-category *scatterplots* is very limited and describes only one technique: the *scatterplot.*

Finally, the only practical and complete example of their taxonomy in use was the classification of techniques implemented by their system, called *XGobi.* Apart from that the techniques mentioned in their work were classified only in respect to the *rendering* category (i.e. scatterplots, traces, and glyphs sub-categories). It is quite difficult to evaluate the taxonomy's usefulness in understanding visualization methods because the authors did not provide sufficient examples nor any comparative observation on those techniques classified according to their taxonomy.

2.2.3 Task by Data Type Taxonomy by Shneiderman

In 1996 Shneiderman introduced the Task by (Data) Type Taxonomy (TTT) for information visualization [176]. The taxonomy was motivated by a basic principle the author called the *Visual Information Seeking Mantra*: overview first, zoom and filter, then details-on-demand.

The taxonomy considers the methods under two aspects, the data type and the task-domain information objects (i.e. generalizations of typical problems that the visualization is trying to solve). The data, in this context, is seen as a collection of items with multiple attributes, classified into seven types: *1D, 2D, 3D, temporal, multidimensional*[8]*, tree,* and *network data*. There are only seven basic tasks described: *overview, zoom, filter, details-on-demand, relate, history,* and *extract*. However, the author suggests that the next natural step is to extend this list.

Based on these two aspects – data type and task-domain – the TTT taxonomy is constructed, and techniques are organized as shown in Table 2.3.

2.2.3.1 Critique

The sub-categories of the TTT taxonomy were designed in such a way as to cover most of the InfoVis field. This makes it useful to understand the area, on the other hand this broad coverage is not sufficiently detailed for the high-dimensional subfield we are interested in. Most of the high-dimensional methods fall into the *multidimensional* category, whereas other complex abstract spaces such as trees, graphs, and networks are classified into a category of their own.

One problem with this particular application of the taxonomy is that it does not distinguish between techniques and software systems (e.g *Spotfire*, a software system, is classified together with *3D scatterplot*, a method). In contrast, classifying exclusively the techniques enables us to consider the methodology behind a technique, abstracting the idiosyncrasies of a particular software system; in turn, understanding the methodology is far more important than understanding a system that implements it because (a) it is at the algorithmic level that significant improvements to a technique can be achieved; (b) software systems usually have a fairly limited lifetime compared to the methods themselves; and, (c) not all the systems are freely available for use or testing, while a technique normally becomes available to the whole scientific community once it has been published.

[8] According to the terminology we have adopted in this thesis this data type should actually be called *multivariate*, however we have kept the original name as it appears in the paper.

[9] Basic tasks comprehends counting, filtering, and details-on-demand.

Data		**Objectives or Tasks**	**Examples**
Type	*Example*		
1D	Textual documents; program source code; list of names	Find a number of items; see an item having certain attributes; see an item w/ all its attributes	Bifocal Lens; SeeSoft; Value Bars; Document Lens; Information Mural
2D	Geographical maps; fbor maps; newspaper layout	Find adjacent items; containment of one item by another; path between items; basic tasks[9]	GIS; spatial display of document collections
3D	Molecules; human body; buildings	Adjacency, above/below, and inside/outside relationships; basic tasks	Human project; 3D trees; Networks; WebBook
Temporal	Time-lines of medical records, project management, historical presentation	Find all events before, after, or during some period or moment; basic tasks	Project management tools; Perspective wall; LifeLines; Macromedia Director
Multidimensional	Relational and statistical databases	Find patterns, clusters, correlations among pairs of variables, gaps, and outliers	3D scatterplots; parallel coordinates; HomeFinder; FilmFinder; Aggregate Manipulator; Spotfi re; Movable Filters; Selective Dynamic Manipulator; VisDB; TableLens; Influence Explorer
Tree	Hierarchical data	Identify structural properties	Treemap; Cone and Cam Trees; TreeBrowser; Hyperbolic trees
Network	Cases for which the relationships among items cannot be conveniently described with a tree structure	Shortest or least costly paths; traverse the entire network	Netmaps

Table 2.3: Summarizing the TTT taxonomy with methods and systems examples.

Our final observation is that no consideration has been made concerning interaction methods such as *brushing and linking*, nor data analysis tasks such as *multidimensional scaling* or *principal component analysis.*

2.2.4 Classification Scheme by Wong and Bergeron

The work of Wong and Bergeron in 1997 [212] provided a historical overview of the visualization field focusing on high-dimensional visualization. They have traced the field's origin back to a period before 1976 when most of the visualization work was done by

mathematicians, physicists, and statisticians, supported by psychologists. Then they describe three periods of intense development of the field leading to the present day.

The major focus of the paper, however, is on the last two periods of the development, which started in 1987 with the publication of the seminal paper by McCormick *et al.* [136]. During those periods several techniques, and systems were developed. They noted that the main objectives of the high-dimensional visualization methods are: to visually summarize the data, and to find key trends and relationship among variates.

The authors recognize the difficulty in finding a suitable set of criteria to properly categorize high-dimensional visualization techniques. They suggest, however, possible candidates such as the goal of the visualization, the type and/or dimensionality of the data, and the visual dimensionality of the technique.

They group techniques in three major categories, *techniques based on 2-variate displays*, *multivariate displays*, and *animation*. Because the characterizing feature of these categories is based on the visual nature of a technique we may conclude that the authors introduced a display-based classification scheme. The three categories are briefly described next.

2.2.4.1 Techniques based on 2-variate displays

As the name suggests this category encompasses the techniques whose visual representation is essentially two-dimensional. Techniques in this category were designed to deal with just a few hundred data items, and they rarely use colour to depict information.

The origin of techniques that fall into this category is closely related to the field of Statistics, and they have hugely been influenced by work done by Tukey [69, 197], Tufte [195, 196], Cleveland [47–49], and Chambers [39]. Consequently, it is noticeable that the main concern of techniques in this category is to show correlation between variates and provide an exploratory tool that affords the identification of models that better describe the data.

Most of the methods here are made of points and lines and the technique most representative for this category is unquestionably the *scatterplot matrix*. They also classify in this category a list of auxiliary tools and concepts that the techniques in this category normally use or follow, such as: using a reference grid to help locating data items; displaying a fitting curve in an attempt to describe a data model; and employing the *banking*[10] principle to improve the visual perception of the plot.

[10] A principle that requires the adjustment of the aspect ratio to improve the perception of the orientation of line segments in a graph [50].

2.2.4.2 Multivariate displays techniques

This is the group where the majority of methods belong. The determining features of this category are: the output of coloured and relatively complex images (which usually means a steep learning curve for the users); a high-speed generation of display to support a considerable degree of interaction; and finally, the ability to deal with datasets more complex than the ones tackled by techniques in the previous category.

This category is further divided into sub-categories[11], namely *brushing*, *panel matrix*, *iconography*, *hierarchical displays*, and *non-Cartesian displays*.

Brushing This sub-category has only one element: *brushing* applied to a *scatterplot matrix*. *Brushing* is a mechanism that affords direct manipulation of the *scatterplot matrix*'s visual display. According to the authors *brushing* can be of two kinds, either *labeling* or *enhanced linking*. The former happens when one causes information label(s) to pop-up once an item has been 'brushed' by an interactive device, such as a mouse pointer; whereas the latter works in a way similar to that described earlier in Section 2.2.2.2.

Panel matrix This sub-category describes techniques that represent high-dimensional visualization as an array of 2D plots, generated by pairwise combination of variables.

Iconography The defining criterion for techniques in this sub-category is to use graphical objects such as *glyphs* or *icons* to represent data. The data attributes or variables are mapped to geometric features of these graphic objects. Normally the number of graphical objects is equal to the cardinality of the high-dimensional dataset and they are arranged on the display in such a way as to reveal visual patterns recognizable by humans. It is expected that these patterns represent interesting behaviour of the data.

Hierarchical displays The common characteristic of techniques in this sub-category is to impose a hierarchical organization upon the *visual space*. Then subsets of the variables are assigned to different levels of that hierarchy, until all the dimensions have been used. Commonly the dimensionality of each hierarchical level is ≤ 3.

Non-Cartesian displays In this sub-category the techniques rearrange the axes to be non-orthogonal and the data is displayed along the modified axes.

[11] We have adopted the term *sub-category* instead of the original *sub-group* to keep a consistent terminology.

2.2.4.3 Animation based techniques

This is the third category of the taxonomy and comprises the methods that use animation to enhance the presentation of the data. The authors did not introduce any sub-category for this group.

2.2.4.4 Critique

Firstly we present Table 2.4 that summarizes the Wong and Bergeron classification of concepts, tools, techniques, and software systems.

Group 1: Techniques Based on 2-variate Displays
Reference grid, fi tted curve, banking, scatterplot matrix

Group 2: Multivariate Based Techniques				
Brushing	*Panel matrix*	*Iconography*	*Non-Cartesian displays*	*Hierarchical display*
Brushing technique	Hyperslice; hyperbox	Stick fi gure; autoglyph; color icon	Parallel coordinates; VisDB	Hierarchical axis; dimension stacking; worlds within worlds
XmdvTool[12]				

Group 3: Animation Based Techniques
Grand tour, Exvis, scalar visualization animation model

Table 2.4: Listing the techniques and methods classified according to the three categories of Wong and Bergeron's taxonomy.

The three major contributions made by the authors are the following:-

1. They have contributed to the clarification of terms such as *multivariate* and *multi-dimensional*, providing a more formal definition for those terms. They have also presented a historic overview, contextualizing the evolution of the field over a 30 year period (1977-1997).

2. They have compiled a concise and useful description of several representative techniques, concepts, and software systems related to the multivariate multidimensional visualization field.

3. They have introduced a classification scheme for multivariate multidimensional visualization techniques. The proposal of credible classification schemes is always

a welcome contribution to the field, although we believe that their classification presents some shortcomings we discuss below.

The first shortcoming is that they do not distinguish interaction techniques (e.g. *brushing*), concepts (e.g. *banking*), tools (e.g. *reference grid*), visual techniques (e.g. *scatterplot matrix*), or software systems (e.g. *XmdvTool*). As a consequence one could argue, for instance, that the *Exvis* fits into the *Iconography* sub-category, and the animation is just an interaction feature, rather than a defining one. The second point is the creation of the *brushing* sub-category that describes only one technique. The third drawback is that the definition of the category for *techniques based on 2-variate displays* may overlap with the definition of the sub-category *non-Cartesian displays* because the *parallel coordinates* technique, for example, fits into both definitions. Finally, the categories of their classification do not account for data analysis tasks.

The authors recognize, however, that finding a convincing set of criteria that clearly differentiate the visualization techniques is a difficult task.

2.2.5 Classification Scheme by Keim

In 2000 Keim presented an overview and a classification scheme for high-dimensional visualizing techniques [108]; this was later updated in [109]. Keim has defended the need for a formalization of the field aiming at (1) a better understanding of the existing techniques; (b) a systematic development of new techniques; and, (c) a formal way to evaluate them.

Keim's classification scheme is based on three aspects: *visualization technique*, *interaction technique*, and *distortion technique*. Each one of the criteria is mapped onto one of three orthogonal axes. These three axes define a three-dimensional classification space. The *X* axis represents enumerated values for interaction methods: mapping, projection, filtering, link & brushing, and zoom. The *Y* axis contains enumerated values for classes of visualization techniques: graph-based, hierarchical, pixel-oriented, icon-based, and geometric. Finally the *Z* axis accommodates two enumerated values for distortion methods: simple and complex.

According to this classification scheme each visualization technique can have any of the interaction methods, used in conjunction with any of the distortion techniques. The interaction techniques allow dynamic changes of the data towards the visualization objective, as well as relating and combining multiple independent visualizations. Table 2.5 shows how some techniques fit into Keim's taxonomy.

[12]The XmdvTool integrates techniques from four different sub-categories.

Visualization Techniques				
Geometric	*Icon-based*	*Pixel-oriented*	*Hierarchical*	*Graph-based*
Scatterplot matrix and coplots; landscapes; projection views; hyperslice; Andrews curves; and parallel coordinates	Stick figures; shape-coding; and color icons	Spiral; recursive pattern; and circle segments	Dimension stacking; treemap; and cone-trees	Cluster-optimized, symmetric-optimized, and hierarchical graph visualizations

Interaction Techniques				
Mapping	*Projection*	*Filtering*	*Zooming*	*Linking/Brushing*
AutoVisual	Grand Tour	Data Visualization Sliders, Dynamic Queries with Movable Filters	IVEE, Pad++	XmdvTool

Distortion Techniques (Simple or Complex)
Perspective wall, bifocal lens, table lens, fisheye view, hyperbolic tree, hyperbox

Table 2.5: Distribution of techniques according to Keim's classification scheme. Note that the author has not placed any technique into either *simple* or *complex* sub-categories of the *distortion techniques* main category.

2.2.5.1 Critique

The author reinforces the need for a more formal basis to describe techniques and, hence, his major contribution was the formal definition of the design goals for *pixel-oriented methods* as optimization problems. The design parameters being optimized are: the pixel arrangement within a sub-window, the shape of the sub-windows, and the order of the sub-windows.

Another positive aspect is a brief qualitative comparison of some techniques. Keim gives to each technique a score (degree scale: very bad, bad, neutral, good, very good) based on his subjective judgment of the techniques on three criteria: *data characteristics* (number of attributes, number of data objects, suitability for categorical data); *task characteristics* (clustering, multivariate hot spots); and, *visualization characteristics* (visual overlap and learning curve). Although the evaluation is entirely subjective it may be a good starting point for a more formal evaluation proposal.

We believe the main contribution of this classification scheme is the distinction between visual representation and interaction techniques. However, the introduction of a third category, distortion techniques, seems artificial and inappropriate. In our view distortion techniques can be seen as an interactive aspect of a visualization technique, one that alters the way data is presented. This flaw was later corrected in the second version of the classification scheme [109]. In that version the three orthogonal criteria axes now correspond to the following categories: *data type to be visualized*, *visualization technique*, and *interaction and distortion technique*. Hence the interaction and distortion categories are now unified and a new category, data type to be visualized, was introduced.

Another problem is that the author has not provided a formal definition for any category. Instead he has defined a category by giving examples of techniques that belong to that category. This may lead to the misinterpretation of a category's characteristics. For example, the *grand tour* has been classified under the *projection* sub-category, however one may argue that this technique should be also classified under the *geometric* sub-category because it relies on geometric projections to present data. Perhaps what the author meant was that the *grand tour* relies heavily on dynamic selection of projections to convey information.

Once more the visualization techniques have not been classified according to all three categories. Rather the techniques have been assigned to individual categories, as shown in Table 2.5. We would expect, for example, that the *grand tour* technique would feature in both *geometric* and *projection* sub-categories, with no representation in the *distortion* axis. Another inconsistency: even though *filtering* is a key feature of the *hyperslice* technique, *hyperslice* has been placed only under the *geometric* sub-category.

Finally, there is no distinction between the visualization techniques and the systems that implement those techniques.

2.3 The Three Stage Visualization Ontology

In this section we introduce our process-based ontology for high-dimensional visualization techniques, called Three Stage Visualization (TSV). We have chosen a process-based approach because the task of visualizing a high-dimensional dataset can be satisfactorily described by three separate stages, which constitute the main categories of our classification scheme.

The examination of the previous classification schemes has enabled us to identify the following shortcomings:-

- Providing categories that describe quite well the entity we wish to visualize but lack details (or categories) that help understand the methodology used by the high-dimensional visualization methods (e.g. *E* notation).

- Failing to define categories that clearly differentiate the visual representation from interaction features (e.g. Wong and Bergeron), or creating categories whose qualities are not clearly defined (e.g. Keim).

- Emphasizing exclusively either the visual (e.g. Grinstein *et al.*) or the interactive (e.g. Buja *et al.*) aspects of a method, while the other aspects receive little or no attention at all.

- Defining a general classification scheme that is not focused exclusively on high-dimensional visualization techniques (e.g. TTT taxonomy).

- Ignoring data analysis methods which are normally used to reduce the dimensionality of an *n*-dimensional dataset, thus an important asset in designing an effective visualization technique.

We have identified two particularly interesting points from the other proposals: the introduction of abstract categories describing the interaction aspect of techniques, done by Buja *et al.*'s taxonomy; and, the separation between visual and interaction aspects introduced by Keim's classification scheme. The first point is relevant because the definition of the interaction categories in abstract terms made it possible to trace a parallel between tasks from EDA and elements of Gestalt theory, which, in turn, has contributed towards an improved understanding of the manipulation aspects of a visualization technique. The second point, the separation between visual representation and manipulation, is important because by introducing dedicated categories for each aspect one can separately analyze either the perceptual issues of the visual presentation or the interaction capabilities of a technique. The separation also favours the comparison of techniques in terms of their visual representation and manipulation aspects.

Our proposal, therefore, tries to make a clear distinction between three different aspects of a high-dimensional visualization method: data analysis, visualization, and interaction. They correspond to the stages that an *n*-dimensional dataset may undergo to be visualized, and we use them as the major classification criterion of our scheme.

In fact we have decided to use the term ontology because its definition best describes what we are proposing. According to *The Free On-line Dictionary of Computing* [95], *ontology* when related to information science is defined as:

> The hierarchical structuring of knowledge about things by subcategorizing them according to their essential (or at least relevant and/or cognitive) qualities.

We shall think of a technique as a process composed of three stages: *data analysis*, *data picturing*, and *data interaction*. Each of these stages corresponds to a category at the highest level of the ontology's hierarchical structure. They are investigated next and whenever pertinent we identify their sub-categories.

2.3.1 The *Data Analysis* Stage

The first classification is done after identifying whether a technique requires any mathematical pre-processing operation upon the data, prior to the visual presentation.

Usually techniques that employ any mathematical operation try to reduce the dimensionality of the data to a more manageable dimensionality or unveil information hidden in the complexity of an *n*-dimensional space. A data analysis task may achieve this in several ways:-

- Finding a low-dimensional coordinate system that preserves the relative distances between data items (*multidimensional scaling methods, MDS*);
- Finding an orthonormal subspace that optimally represents data correlations within a multivariate distribution (*principal component analysis, PCA*);
- Grouping data items that are considered close to one another or grouping similar dimensions following some criteria (*cluster analysis*);
- Finding interesting low-dimensional projections of multivariate data that best describe possible clusters of data items, by optimizing a projection index (*projection pursuit methods, PP*);
- Organizing the patterns within the data into a topological structure that preserves the relation between those patterns using artificial intelligence methods (*neural network algorithms*);

Other common data analysis tasks that do not aim at reducing the data dimensionality but at transforming data for visualization are:-

- Employing simple statistical procedures to better describe the data (*mean, standard deviation*, etc.);

- Imposing a structure on the data based on an empirical model of the underlying entity represented by the data (*interpolation*);
- Preparing the data for the visualization (*re-sampling*, *dimension ordering*, or *mapping categorical data to numerical representation*).

After determining the type of mathematical methods a technique may use, it is necessary to identify what is the technique's main strategy to present data. This corresponds to the classification under the *data picturing* category.

2.3.2 The *Data Picturing* Stage

The *data picturing* category has four sub-categories, each one of them representing the *main*[13] visual strategy that a visualization technique may adopt in presenting the high-dimensional dataset on a lower dimensional display. This category is also the most important of all three categories because it characterizes the essential quality of a visualization method. The sub-categories are:-

- **Filtering:** This sub-category represents techniques that either (a) select a small group of dimensions (usually less than three) from the original set of dimensions, or (b) filter data items based on some restriction imposed on them. Following that a technique presents the results of this filtering process, usually a low-dimensional subset, using any standard low-dimensional technique such as line graph, contour lines, or a scatterplot.

  Most of the methods that adopt the *filtering* strategy diverge in the way they select the subsets, depending on whether the dataset is multidimensional or multivariate. Methods in this sub-category may also show multiple views of the data simultaneously, each encompassing a different subset of the variables. For example, *hyperslice* and *hyperbox* implement a filtering action by providing a simultaneous view of all possible pairwise combinations of dimensions, thus following strategy (a), described above. On the other hand, *data visualization sliders* and *dynamic queries*, use sliders as a tool to select the data items portrayed on screen and offer a single view of the result, thus following the strategy (b).

  Other examples of techniques in this category are *3D scatterplot matrix*, which selects three variables from the original *n*-dimensional dataset; *table lens*, which relies

[13] We explain why we have used the term *main* visual strategy after the description of the four sub-categories.

on a distortion mechanism to integrate the detailed view of selected data items with less detailed graphical representation of the whole dataset; and, *visualization for multidimensional function by projections (VMFP)*, which uses an approximation step called *uniformly distributed sequence* to transform a multidimensional function into a multivariate dataset and then select two or three dimensions to be visualized as 2D and 3D scatterplots, respectively.

- **Embedding:** The techniques in this sub-category organize the data in a hierarchy of dimensions. The methods are differentiable from one another by the algorithm they use to create this hierarchy. For example, for the *dimension stacking* technique the user selects the order in which to embed one coordinate system within another. Firstly the user chooses two variables to form the outermost level of coordinate system (top of the hierarchy). This two-dimensional coordinate system is divided into rectangular bins and within the bins the next two variables are used to span the second level coordinate system. Then the process is repeated until there is only one or two variable left (bottom of the hierarchy). At this point the data is displayed using any suitable visualization method for lower dimensional data. This is, in essence, the same strategy followed by *hierarchical axis* and *worlds within worlds.*

  Alternatively the hierarchy of dimensions is automatically created, as in the *quad-tree mapping (QTM)* technique. This technique deals with an n-dimensional multivariate dataset representing fields of individuals and their corresponding fitness level. These individuals are the result of a genetic algorithm in which they represent alternative solutions to the target problem. The n fields of a data item are transformed into an n-bit binary value. The method locates each data item on a two-dimensional area following this strategy: (1) take the first two more significant bits and use their values to decide which one of the four quadrants of the 2D display area the data item should be placed in – this is the first level of the hierarchy; (2) recursively repeat this procedure, using, in each interaction, the next two most significant bits to guide further subdivision of each quadrant and define the new location for the data item; and, (3) when all bits have been used we reach the last level of the hierarchy, thus the definitive location of a data item. Then we use the fitness value to create a height field.

  Finally, a technique in this category may simply represent a hierarchy inherent to the dataset, as is the case for the *cone trees* technique. This method uses a three-dimensional tree structure as a metaphor to represent the structure of, say, a file system. The technique relies on interactive methods such as rotation, zoom and

panning to overcome the natural problem of occlusion as a result of the three-dimensional representation of the tree on a two-dimensional display. The *treemaps* technique also tries to represent the inherent hierarchy of a dataset, but instead of a three-dimensional representation it applies a strategy similar to that of the *QTM* technique. The difference, though, is that the recursive subdivision of the two-dimensional display area is guided by the number of elements in each level of the dataset's hierarchy, rather than being fixed to four subdivisions as in the *QTM* technique.

- **Mapping:** The essence of any technique in this sub-category is to map the attributes of each individual data item to the *graphical properties* of a *visual mark* (c.f. Chapter 1, Section 1.1.2). A visual mark may be a pixel (e.g. *pixel-oriented, natural textures, circle segments, heat maps, survey plots*), in this case the variates are associated with specific regions on the display, each variate value is assigned to a coloured pixel or texture, and those pixels/textures that belong to the same variable are placed together in the corresponding region; or, an icon (e.g. *stick figure, star glyph, color icons, shape coding,* and *Chernoff faces*), in which case a technique should provide an arrangement of all the icons generated in the mapping step in such a way as to reveal new information about the data.

  Common to all techniques in this category is their intention of exploiting human perception ability to reveal inherent behaviour of a dataset and recognize relationships among data elements.

- **Projection:** The projection sub-category comprises techniques that use any geometric projection of the *n*-dimensional data down to a lower dimensional subspace, usually two-dimensional. A projection, in this case, may be a simple parallel projection on a low dimensional plane (e.g. *grand tour*, and *prosections*); a non-linear projection (e.g. *SOM maps,* and *RadViz*); a projection to a mathematical domain (e.g. *Andrews curves*), or; simply a rearrangement of the axes to be non-orthogonal and display the data along all the axes simultaneously (e.g. *parallel coordinates, multi-line graphs, polar charts,* and *star coordinates*).

At the introduction of the *data picturing* category we mentioned that this classification represents the *main* visualization strategy a technique may adopt. However, a technique does not necessarily have to employ only a single strategy to generate a visualization. Sometimes a technique may use a secondary strategy to accomplish the complete representation of the data. Nonetheless, in our classification we consider only the core or

primary strategy of the visualization process. For instance, the *hyperslice* technique follows the *filtering* strategy to extract subspaces, but uses a gradient field to represent a subset on screen. Therefore the *hyperslice*'s *data picturing* stage is essentially classified in the *filtering* category, while a secondary classification would place *hyperslice* in the *mapping* category.

Table 2.6 summarizes the classification of the visualization methods mentioned earlier under the *data picturing* category, considering only the main classification and ignoring any secondary strategy a method may use. Note that the *projection* approach is the most popular having ten instances, followed by *mapping* with nine, *filtering* with eight, and *embedding* approach with only five instances from our list. Also observe that the number of techniques in each of the four strategies is very close to the average (8), thus a balanced distribution of techniques throughout the categories. A final remark is that the *data picturing* category could still be further subdivided, according to the different algorithms a technique may follow to produce a visualization.

2.3.3 The *Data Interaction* Stage

This is the final stage of the classification. The *data interaction* category is representative of the interactive methods used to improve the visualization process. Examples of interaction mechanisms in this category are:-

- *Aggregation*: This mechanism can be realized at two levels: data item level, or variate level. The *hierarchical parallel coordinates* technique uses aggregation at the data item level, allowing the user to interactively modify parameters that control the aggregation of data items into clusters.

  The *Visual Hierarchical Dimension Reduction (VHDR)* technique implements aggregation at the variate level. In this technique the variates are placed into clusters and a representative variate is selected (either the 'centre' dimension of the cluster, or a new variate which is an average of those in the cluster). The user can change the aggregation by modifying the variate hierarchy within a cluster, or by selecting different levels of detail to present the clusters.

  Note that *aggregation* in these examples are, initially, defined automatically, then it becomes available for user interaction.

- *Rotation*: This can either be *rotation in the n-dimensional data space* used for exploration of better projections into low dimensional spaces, or rotation executed in the visual space (e.g. rotations of *3D scatterplots*);

Technique	**Data Picturing Category**			
	Filtering	*Embedding*	*Mapping*	*Projection*
Andrews curves				x
Autoglyph			x	
Chernoff faces			x	
Circle segments			x	
Color icon			x	
Cone trees	x			
Data Image			x	
Data Visualization Sliders	x			
Dimension stacking		x		
Dynamic queries	x			
Grand Tour				x
Star diagrams				x
Hierarchical axis		x		
Hyperbox	x			
Hyperslice	x			
SOM maps				x
Multi line graph				x
Parallel coordinates				x
Polar charts				x
Prosections				x
QTM		x		
RadViz				x
Scatter plot matrix	x			
Star coordinates				x
Stick fi gure			x	
Survey plots			x	
Table lens	x			
Texture-based			x	
Treemaps		x		
VisDB			x	
VMFP	x			
Worlds within worlds		x		
TOTAL	8	5	9	10

Table 2.6: Classification of high-dimensional techniques under the *data picturing* category of the Three Stage Visualization ontology.

- *Linking & brushing*: This is a powerful dynamic method that improves the exploratory power of techniques based on multiple different projections, such as *scatterplot matrix*. Several scatterplots in a *scatterplot matrix*, for instance, can be 'linked' by shading subsets of points in one projection and highlighting the corresponding points in all other projections. This may facilitate the identification of

correlations and dependencies between variables.

- *Interactive selection*: This mechanism is used in techniques to accomplish browsing of data items or to query entire *n*-dimensional datasets. This involves, for example, the imposition of thresholds to control the visualizable data items (e.g. *data visualization sliders*), or the definition of ranges for each variable (e.g. *prosection matrix*). Examples of techniques that implement this interactive mechanism are discussed later in Section 2.4.3.

- *Zooming & panning*: These are well known methods, present in almost all techniques. The idea here is to allow the user to control the level of detail of the visualization, which can vary from a compressed overview of the dataset to a full detailed view of individual data items. *Panning* allows the user to move over data items within the same display resolution.

- *Interactive mapping*: This mechanism is found in techniques that afford dynamic modification of the variables assigned to the *graphical properties* (e.g. colour, shape, or brightness) of Visual Marks [20, 29].

- *Animation*: This mechanism allows users to control the parameters of an animation designed to enhance the presentation of the data. The *grand tour* technique, for example, animates a sequence of *scatterplots*, trying to show all interesting low-dimensional projections of an *n*-dimensional dataset.

- *Focus+context & distortion*: This interaction method combines subsets of data at a high level of detail with data at a low level of detail. The goal here is to support the users when they execute drill-down operations within the dataset. Good reviews of techniques that implement this mechanism were presented by Leung and Apperley [124], and Carpendale *et al.* [36].

  Note that this sub-category may be regarded differently, depending on the way it is applied in a technique. For example, one may consider it as a visual representation method because it defines how the data is visually presented, whereas others (as ourselves) may classify it as an interaction mechanism, because it only affects the way the data is already being visually presented. There is also a third interpretation in which we think of it as a pre-processing step applied to the data before the visualization stage begins. This was discussed, for example, by Cohen and Brodlie in [52].

- *Details-on-demand*: This mechanism is used when the user desires to obtain detailed information, such as the original values of a data item, from a small selection of data items, usually picked up by a mouse pointer or equivalent device.

Note that the lists of descriptive elements for both *data analysis* and *data interaction* categories are by no means exhaustive ones. In the future these lists should be improved either by adding more elements to each category; or, preferably, by replacing the instances with a smaller but more representative set of sub-categories, following a similar organization used to describe the *data picturing* category.

2.3.4 Classifying Some Examples using TSV

Hopefully some examples should make the process of classifying techniques under TSV more clear. Table 2.7 shows a few techniques classified according to this proposal.

Technique	**Three Stage Visualization**		
	Data analysis	*Data picturing*	*Data interaction*
Andrews curves	Statistical manipulation (mean calculation, standard deviation); PCA	Projection	Interactive mapping (experimenting with different variates as function parameters); interactive selection (selecting data item to be transformed into function plottings)
Autoglyph	Statistical manipulation (standard deviation)	Mapping	Interactive mapping (changing mapping of colours)
Chernoff faces	Clustering	Mapping	Interactive mapping (changing mapping of face attributes)
Color icon	-	Mapping	Interactive mapping (changing mapping of color icon parameters)
Cone trees	-	Filtering	Animation; zoom + pan; details-on-demand (ability to collapse the tree structure if desired); interactive selection (search of elements is possible via pop-up property sheet); focus+context & distortion (the three structure gives an overview of the data and clicking on one item brings it to user attention)
Data Image	Sorting data items; Calculate distances between observations using different metrics	Mapping	Interactive mapping (changing the order of the dimensions)
Dimension stacking	Normalization; sampling	Embedding	Interactive mapping (defi ning ordering of dimensions to form the hierarchy); defi ning size of the bins
Grand Tour	Projection pursuit	Projection	Animation (choosing the path for the tour)
Hyperbox	-	Filtering	Distortion; interactive mapping (assigning dimensions to edges of the hyperbox); interactive selection (slicing the hyperbox along a selected dimension to obtain new hyperboxes of reduced dimensionality by collapsing the sliced hyperbox along the cut direction)
Hyperslice	Interpolation; fi nding local maxima/minima	Filtering	Zoom and pan (Changing the center point, defi ning paths in *n*-dimensional space); linking
Parallel coordinates	Normalization; PCA; clustering	Projection	Choosing the order of the dimensions/axes; brushing + linking
Table lens	-	Filtering	Focus+context; zoom + pan; details-on-demand by selecting a column
Worlds within worlds	Interpolation (depending on the dataset type)	Embedding	Virtual reality interaction; zoom + pan; rotation; choosing the hierarchy for the dimensions

Table 2.7: Classification of high-dimensional techniques under the TSV ontology.

These are some observations based on the classification shown in Table 2.7:-

1. The *cone tree* and *table lens* techniques (both of them from the InfoVis application realm) have no *data analysis* stage but, on the other hand, these methods have several elements under the *data interaction* category. Perhaps the reason for the absence of a *data analysis* stage is that these methods are not primarily designed to deal with numerical data but rather with categorical data. Probably the strong interaction side of these methods is influenced by InfoVis guidelines, more concerned with user interface and interaction capabilities than the techniques from the SciVis field.

2. Methods in the *embedding* class usually benefit from an interaction ability to change the order in which the dimensions are hierarchically organized.

3. Only two methods (*grand tour* and *cone trees*) have made use of animation to improve the visualization process. This is a powerful interaction feature that has been systematically under-used.

4. Normalization[14] is a fairly common step for techniques that deal with numerical data.

5. All methods classified in the *mapping* category deal primarily with multivariate datasets.

2.3.4.1 Critique

We have designed the TSV ontology trying to address the several shortcomings listed at the beginning of this section (c.f. Section 2.3). The scope of our proposal is clearly the high-dimensional visualization techniques in both SciVis and InfoVis, with more emphasis on those techniques that deal with numeric, or categorical data that can be mapped to a numeric representation without compromising the outcome of the visualization.

We have tried to clearly identify a set of criteria for the main categories and sub-categories, as well. We have based them on the stages that a dataset may undergo to be visualized, using abstract definition whenever possible (e.g. the three main categories, and the *data picturing*'s sub-categories). Note that our scheme possesses a category for mathematical data analysis methods, although one may argue that its sub-categories are not well defined.

[14] Normalization in this context means mapping the numeric domain of a variable into the range $[0, 1]$.

We have noticed that when techniques are fully classified, as in Table 2.7, it is relatively easy to compare their features which may lead to conclusions such as those enumerated in the previous section. During the process of categorizing the techniques we focused on the methodology behind a technique, rather than any particular implementations.

Finally we have tried to give the same level of attention to all stages, providing a formal definition for each category. However, the *data picturing* category has received more attention than the others for being, in our opinion, the defining quality of a technique.

2.4 Reviewing Relevant Techniques

This section reviews in more detail some visualization methods and multiple views coordination. The criteria for selection of the reviewed techniques were: to follow the *filtering* strategy; to be relevant for the visualization of both multivariate and multidimensional data; and, to have influenced our suggested visualization technique.

We have chosen to review only methods from the *filtering* sub-category because this is the approach we have adopted in our proposal for a common framework, described in the next chapter; the multiple views coordination is also reviewed here because we rely on the use of several views to present the 'filtered' data. Whenever pertinent examples are given to illustrate the techniques. We were specially interested in recognizing their strongest features, understanding some of their limitations, and identifying possible improvements that could be done to enhance their visualization power.

2.4.1 Why *Filtering* strategy?

According to Spence [181] the mind is not adept at processing large amounts of information but prefers to simplify complex information into patterns and easily understood configurations. Therefore we may hypothesize that the *filtering* strategy represents a valuable approach since it advocates the presentation of the data in smaller subsets, rather than trying to show all of it in a single view, as in the *mapping* approach.

Using subsets with low-dimensionality has also the advantage that we can rely primarily on spatial mapping (i.e. map the data dimensions to the spatial dimensions of the display) to visually present the data. According to Card *et al.* [32, Chapter 1, page 26], "the most fundamental aspect of a Visual Structure is its use of space", which is backed by the fact the space is a perceptually dominant feature. This is why spatial positioning is the most important encoding device for a visual representation, as suggested

by several perceptual guidelines by Cleveland [47], Rheingans and Landreth [159], and Tufte [195, 196].

This hypothesis is also fostered by two facts: (1) sometimes a high-dimensional phenomenon can actually be governed (thus best described) by a few simple variables (called "hidden causes" or "latent variables") [129], hence the investigation of a reduced set of dimensions at a time might be helpful in finding hidden causes for a given phenomenon; and, (2) each other strategy results in the distortion of the *n*-dimensional relationship between data points in order to map the data to the display, whereas filtering-based techniques explicitly attempt to preserve these relationships [37].

Here is a list of further arguments found in the literature that support our choice of the *filtering* strategy:-

- According to Yang *et al.* [217] because *dimension filtering* removes some of the variables from the display it is essential for visualizing a high-dimensional dataset. The reasoning is that none of the existing high-dimensional visualization techniques can map all variables at the same time without cluttering the display.

  Yang *et al.* also argue that *dimension filtering* is more intuitive to the user than, say, approaches that rearrange the variables in a way that they have little intuitive meaning to the users (e.g. a *scatterplot* of a multivariate dataset whose dimensionality has been reduced by a *MDS* algorithm). They also emphasize that *dimension filtering* is "more flexible in that it allows users to interactively select or unselect dimensions to be filtered."

- Buja *et al.* [30] affirm that attempts at showing all the information at the same time (which they called dense encoding) are rarely successful. They also stressed that "it is usually more effective to construct a number of simple, easy to understand displays, each focused clearly on a particular aspect of the underlying data."

- Grinstein *et al.* [84] defend a strategy they call *comparative visualization*. It encourages the creation of low-dimensional subspaces by selecting variables, which, in turn, should be coupled with the ability to dynamically select arbitrary subsets from these lower-dimensional subspaces. This coupled action, according to Grinstein *et al.*, is "the basis for the examination of relationship in the data."

- Finally, Keim [109] declares that in exploring large datasets, it is important to interactively partition the dataset into segments and focus on interesting subspaces, which he calls *interactive filtering*.

The main disadvantage of this strategy, however, is that this method is constrained by human processing limitations in concentration and memory. In addition to that one may argue that because the *filtering* strategy may require the use of multiple views it would increase the cognitive load on the user because they would have to learn how to integrate the 'pieces' to get an idea of the high-dimensional entity being visualized. Although these two points may represent a limitation to the *filtering* approach we believe that its numerous advantages, mentioned earlier, outweighs any cognitive constraints on the user that may arise from its application.

2.4.2 'Slicing' the *n*-Dimensional Data Space

In the first chapter we have provided some examples of high-dimensional entities and affirmed that they originated from various sources that can be defined as either multidimensional (e.g. mathematical functions, parameter optimization problems); or, multivariate (e.g. statistical measurement, data *census*, stock market, etc.), which are commonly interpreted as point-clouds defined over a high-dimensional space.

In both instances the data structure or behaviour must be analyzed and understood to provide knowledge and insight. A typical strategy adopted by various visualization techniques to visualize such high-dimensional entities is to make several distinct two-dimensional projections of the *n*-dimensional space in which they are defined.

2.4.2.1 Scatterplots matrix

Creating several distinct projections is exactly the approach used in a *scatterplot matrix* when dealing with multivariate data. It is, probably, one of the oldest and most frequently used methods to project multivariate data down to two dimensions [15, 39, 47–49, 197]. It is considered a standard visualization and is available in almost all visualization systems for high-dimensional data [191, 205].

This method generates $n(n-1)/2$ pairwise parallel projections, where n is the dimensionality of the multivariate dataset. Each projection represents the relationships within the data between pairs of variables, and they are arranged in a matrix-like structure that makes it easier for the viewer to quickly recognize the variables associated with each projection.

Figure 2.1 shows an example of a *scatterplot matrix* applied to a four-dimensional Iris dataset [68], containing 150 observations in three clusters[15] (50 observations in each cluster), having four numerical attributes: sepal length, sepal width, petal length, and

[15]The three clusters correspond to the three species of Iris flower: *setosa*, *verginica*, and *versicolor*.

petal width. Note that not all three clusters are visually distinguishable in the *scatterplot matrix* of Figure 2.1.

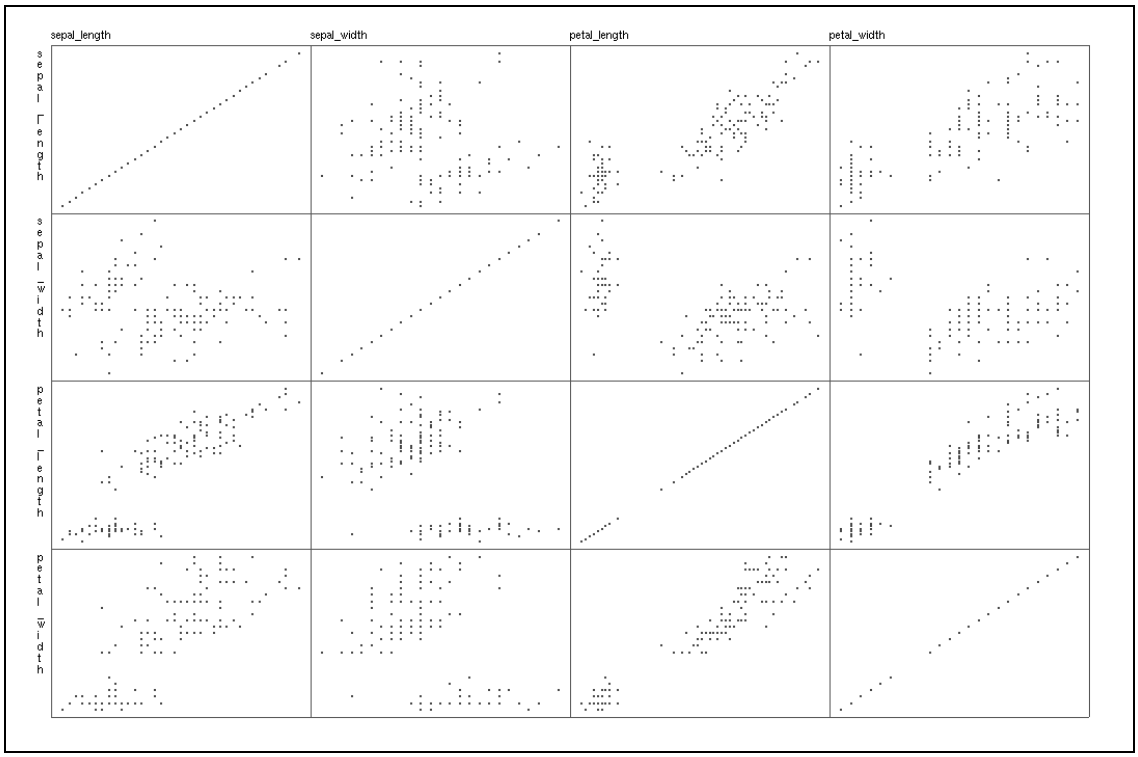


Figure 2.1: Scatterplot matrix of the Iris dataset [68]. Note that two clusters are visible in all panels, but the larger cluster is in fact two clusters very close together. Image generated with XmdvTool [205].

2.4.2.2 Two- and three-dimensional scatterplots

The basic unit of a *scatterplot matrix* is the *two-dimensional scatterplot.* It is a common visualization metaphor for discrete data in which each observation or data item is mapped to a point in a Cartesian coordinate system whose axes are defined by two variates selected from the original set of variates. The range for each axis is normally the same range of the variate associated with it. This arrangement forms a so-called point-cloud representation, which may provide the viewer with a 'picture' of possible correlations between data variates used to form the plot.

Two-dimensional matrix is very efficient in revealing structure and relationships between variates because the technique maps the data variates to the coordinates of points in a graph. This simple presentation strategy enables us to exploit our natural and spontaneous perceptual ability to distinguish clusters of points (groups of related data items) and shapes in space [206].

Three-dimensional scatterplots are a natural extension to the original idea. In this technique an extra axis is added to the *two-dimensional scatterplot*, which introduce an extra variate space. Now points are located in the three-dimensional *visual space*, and then projected onto a 2D display area. The points maintain a 3D appearance because their representation is accomplished by means of depth cue mechanisms, such as interposition (overlapping elements in a scene indicates proximity), familiar size (the brain compares a perceived size with an expected size to ascertain distance), or the use of grid on reference planes (this shows deformation of the grid as we move 'into' the image, thus away from the viewer).

The extension of a scatterplot to three dimensions makes it possible to identify existing correlations involving three variates. On the other hand the three-dimensional representation on a flat display introduces the problem of occlusion, in which data points may cover other data points depending on the viewpoint adopted. To overcome this situation it is necessary to have an interaction interface more complex (e.g. supporting three-dimensional rotations) than its two-dimensional counterpart. Additionally both 2D and 3D *scatterplots* will certainly benefit from an interface that implements 'picking' of data points by means of a pointing device such as a mouse.

Another problem with *three-dimensional scatterplots* arises when the number of points becomes large. The effectiveness in distinguishing points is, therefore, directly dependent on the total number of points and the resolution of the display. One way of addressing this problem was described by Becker in [14]. He proposed the use of binning in conjunction with volume rendering in the following manner: the space of a three-dimensional scatterplot is divided into a three-dimensional grid of bins. Each bin is a voxel whose opacity is a function of the number of data points in that bin and its colour is determined by averaging one of the variates for all data points in that bin.

A final caveat: if the three-dimensional scatterplot is to be presented as a static image, a good practice is to enhance its visual presentation by adding some depth cue device, such as orthogonal line segments connecting the points to any of the three orthogonal planes; or the representation of points as spheres using perspective projection, which helps in identifying location of data points in a three-dimensional space [22].

2.4.2.3 Prosection matrix

A variant of the *scatterplot matrix* method is called *prosection matrix* [198]. It essentially works in the same way as the original *scatterplot matrix* with the added bonus of allowing the user to define the range of each variable. Only observations whose variables lie within their corresponding range (or *section*) is projected on the *scatterplot matrix* (see

Figure 2.2-a). Hence the name prosection, which, in fact, was originally introduced by Furnas and Buja [76].

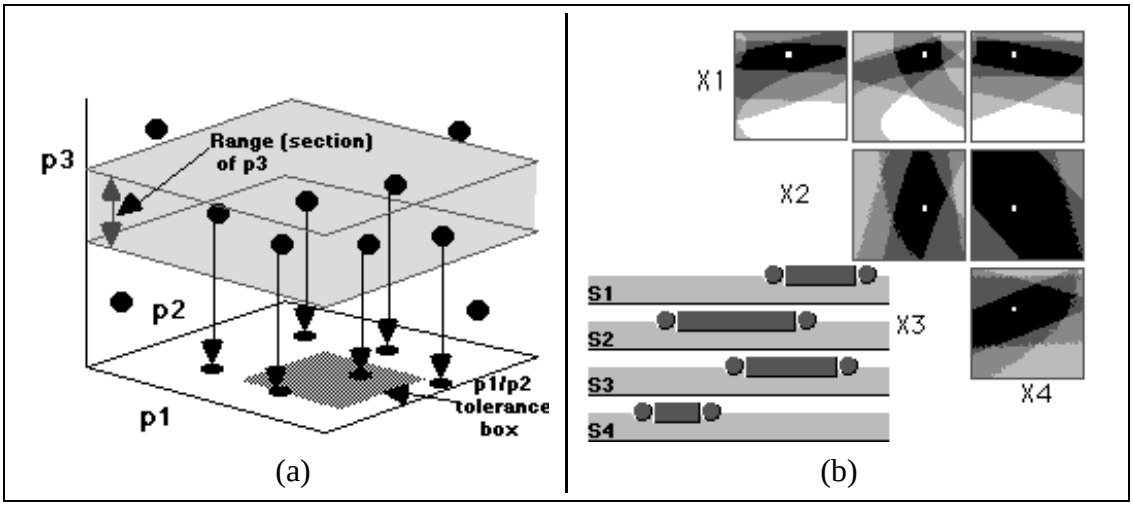


Figure 2.2: Visualization of 3D parameter space with prosection matrix (pictures taken from [198]). Picture (a) shows a section defined on parameter $p3$ and projected down on the plane defined by parameters $p1/p2$. Note the tolerance box as a dark grey square on that plane. Picture (b) shows an actual *prosection matrix* visualization of a four parameter space. In that the different grey areas represent the levels of satisfaction for performance requirements, with the darkest grey level meaning the best satisfaction. In this case the tolerance region is reduced to the white dot in the centre of each scatterplot.

The prosection matrix was originally applied to the exploration of parameter space in a design simulation of lamp bulbs. A design simulation usually requires the definition of acceptable ranges for each parameter of the design and a tolerance range within which a design is considered valid. Note that the parameter space in this case is defined over a continuous range. Thus the data for a single scatterplot are obtained from the design model, by providing the continuous acceptable ranges of the two parameters associated with the scatterplot, while the other parameters are randomly sampled over their corresponding tolerance range. Consequently if the tolerance range is very small all the input parameters of the design model are set to a single point (no random sampling), thus yielding a smooth output; whereas if the tolerance range (which is a square region) is interactively expanded on the prosection matrix, so is the scatterplots' degree of fuzziness due to the random sampling applied on the corresponding increased parameters (see Figure 2.2-b).

2.4.2.4 HyperSlice

Sometimes the high-dimensional object is a function defined over a continuous high-dimensional domain, for which case it is a common approach to project slices of the

function instead of the whole high-dimensional domain. Functions of this nature are called *multidimensional functions* or *scalar functions of many variables* and are frequently used to describe models and simulations in Engineering and Mathematics applications. A typical application example occurs in parameter optimization problems, where we wish to visualize the value of an objective function, $F = (f_1)$ in terms of a large number of control parameters, $X = (x_1, x_2, \ldots x_n)$, say.

In 1993 van Wijk and van Liere proposed a technique called *hyperslice* [199, 200] to tackle this type of function. *Hyperslice* looks at all 2D orthogonal subspaces of X, and presents a grid of contour maps. Each of these subspaces is a slice of the original data obtained by fixing the value of $(n-2)$ parameters to the corresponding coordinates of a focal point, $c = (c_1, c_2, \ldots, c_n)$; and varying the remaining two parameters within a specified region. Thus it reduces from one *n*-dimensional space to *m* 2D spaces, where $m = n(n-1)/2$. In fact the subspace visualizations are laid out in a symmetric *n* x *n* grid, with the diagonal showing *n* 1D visualizations, where only one parameter varies (so a line graph is drawn).

Figure 2.3 shows the *hyperslice* concept applied to a 3D ellipsoid-type function:

$$f(x,y,z) = \frac{(x-1.0)^2}{(0.8)^2} + \frac{(y-1.0)^2}{(0.5)^2} + \frac{(z-1.0)^2}{(0.2)^2}$$

Note how the different focal points of Figure 2.3-b and Figure 2.3-e yield different visualizations of the same function.

Hyperslice's strongest point is the direct relation between screen space and *data space*, which is lost only if *n*-dimensional rotations in *data space* are applied. This relation made it possible to design a user interface that affords direct manipulation of operations such as navigation, identification of *extrema*, and definition of paths in *n*-dimensional space.

Although the authors decided to represent the function visually by two-dimensional slices, they recognised that three-dimensional techniques seem to be the most natural choice for the basic representation of the multidimensional function. The authors argue that the 3D representation has the advantage of encoding as many dimensions as possible simultaneously. Notwithstanding, they decided not to use 3D as the basic representation for the *hyperslice* for three reasons: (1) the techniques for volume rendering, at that time, were too slow for direct manipulation; (2) the interpretation of a 3D representation is more difficult than simpler two- and one-dimensional forms; and, (3) the interaction in 3D is not trivial.

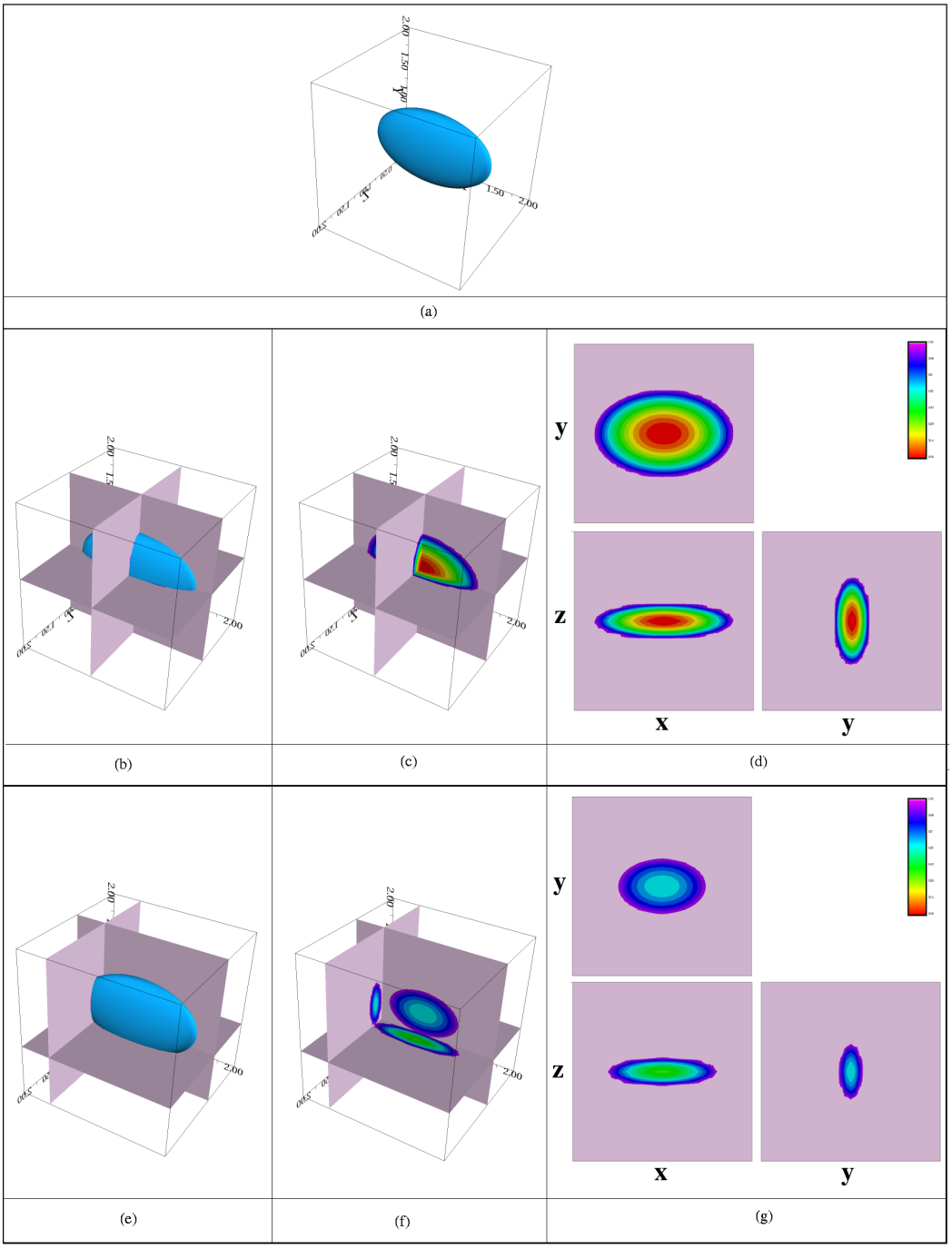


Figure 2.3: *Hyperslice* technique applied to a 3D ellipsoid expressed by the function $f(x,y,z) = \frac{(x-1.0)^2}{(0.8)^2} + \frac{(y-1.0)^2}{(0.5)^2} + \frac{(z-1.0)^2}{(0.2)^2}$, with f restricted to the interval $[0.0, 1.0]$. Picture (a) shows an isosurface $f = 1.0$ of the function. In (b) the focal point is set to $c_1 = (1,1,1)$ located where the three orthogonal slicing planes intersect, which generates the contours in (c). Picture (d) shows a *hyperslice*-type arrangement of those planes. In (e) the focal point was moved to $c_2 = (0.4, 0.7, 0.85)$, generating the contour planes in (f), which in turn, yields the arrangement shown in (g).

Although the *hyperslice* method has proved to be a useful tool in visualizing multidimensional functions, we have identified three shortcomings:-

- Sometimes a single collection of two-dimensional slices is not enough to readily distinguish special features in a function. Instead we need to investigate several of those slices, taken from different locations in the *n*-dimensional space or even navigate through that space to mentally 'visualize' such a feature.

  In contrast if we had a collection of three-dimensional 'slices' we might have identified the same feature (e.g. a *minimum* point) without requiring navigation.

- Because *hyperslice* represents the multidimensional function as a matrix of orthogonal two-dimensional slices simultaneously, all the data corresponding to those subsets must be present before drawing. Thus the method is constrained in terms of performance by both the dimensionality of the function and the number of samples necessary to plot each slice.

- *Hyperslice* provides only a single focal point. Hence the only way of comparing different regions in the *n*-dimensional space is by moving the focal point through a user defined path. However, moving the focal point has an inevitable side effect: the loss of the initial visualization.

2.4.2.5 Worlds within worlds

In 1990 Feiner and Beshers conceived the *worlds within worlds* technique [19, 67]. It is based on the same principle as *dimension stacking*, with a maximum of three variables represented at each level, creating an interactive hierarchy of displays. The user interacts with the system using a data glove to define a position in the space in which a new three-axis 'world' is created to accommodate three more variables (i.e. a new coordinate level of the hierarchy). The process is then repeated until all dimensions have been mapped, ending up with a visualization corresponding to the last variable(s) defined. Different variable mappings yield different views of the data. Note that this placement of subspaces works more as an arrangement of subspaces than really adding visual (quantitative) information to the visualization.

The strongest point of this technique is the direct manipulation of the graph by means of a haptic device such as a data glove. However this is also its downside because such an advanced interface is not as easily available as the common desktop environment, which makes this method not as accessible as other desktop-based techniques.

Another shortcoming is that the arrangement of several low-dimensional slices into a three-dimensional grid causes cluttering of the display. This problem is partially eased by the powerful interactive capabilities but this solution does not scale as the number of dimensions increases.

2.4.3 Dynamically Filtering the *n*-Dimensional Data Space

Another common strategy for techniques in the *filtering* sub-category, particularly frequent in the Information Visualization field, is to perform dynamic queries with immediate visual results.

2.4.3.1 Dynamic query

The *Dynamic Query (DQ)* technique [2] implements the essence of this approach. The technique associates an interface widget, usually a slider, to every variate of a multivariate dataset, defining a query component. These query components act as a filter, reducing the number of items in the filtered subset.

The main advantage of the DQ technique is that it allows rapid, incremental and reversible changes to the query parameter, which is accomplished quite intuitively by simply dragging a slider. This operation affords exploration tasks with immediate feedback, because the result of a slider manipulation done by the user is instantaneously reflected on a visual display, usually a two-dimensional scatterplot of the 'filtered' results.

The individual query components are combined with simple AND logical operations. One of the DQ's disadvantages is that an OR logical operator can only be emulated through a sequence of queries. Also the effect of the DQ is global in the sense that it affects all data items and it cannot be limited to a portion of the data.

The DQ technique has been explored and extended in several ways. For example the *FilmFinder* application follows the *visual information seeking* principle. This principle is based on the DQ approach but improved with tight coupling interface and *starfield visualization* to support browsing [1]. Another study extended DQ by allowing the creation and decomposition of aggregates (groups of data items), which improves the data manipulation capabilities of the original method [80]; whereas Fishkin and Stone [70] tried to solve the DQ's lack of disjunctive queries by encoding each operand as a *magic lens filter* [187]. Finally, Eick in [64] proposed *data visualization sliders* which incorporate the visual representation of the data into the sliders themselves.

2.4.3.2 Infocrystal

Another similar technique is called *infocrystal*, proposed by Spoerri [185]. *Infocrystal* is a visual query language designed to provide all the possible relationships between n concepts. The *infocrystal*'s interface is a two-dimensional iconic representation of a Venn diagram. The disjoint subsets of a Venn diagram are mapped to the *interior icons* whose shape reflect the number of criteria[16] satisfied by their contents. For example, a circle represents one criterion (e.g. a data item d_1 belongs to a set T_1, or $d_1 \in T_1$), a rectangle represents two criteria (e.g. a data item d_1 belongs, simultaneously, to the sets T_1 and T_2, or $d_1 \in T_1 \cap T_2$), a triangle represents three (e.g. $d_1 \in T_1 \cap T_2 \cap T_3$), and so on. A border around these icons contains the *criterion icons* that represent the original sets. Figure 2.4 illustrates this concept for three sets of data.

An *infocrystal* generates $2^n - 1$ disjoint subsets called *constituents*. A Boolean query can be achieved by selecting individual *constituents* or by combining them. Also users can assign relevance weights to each *criterion icon* and control the overall behaviour of an *infocrystal* by changing the acceptable threshold so that only the criteria that are above the threshold value are considered.

The *infocrystal* technique is based on encoding principles to reflect query information. The *interior icons* are shape-coded to reflect the number of criteria that they satisfy; they are placed relatively next to the original sets involved in the Boolean query that they depict; same shaped icons are grouped in 'invisible' concentric circles and the closer an icon is to the centre the higher the number of criteria satisfied; each side of an icon faces an original set involved in the query, and finally; saturation, colour and size are used to code various quantitative information.

Although the *infocrystal* provides a very compact and elegant representation of all possible Boolean queries involving n concepts, it incorporates so many coded information into an icon representation that it may become cognitively overwhelming for the user to comprehend all the relevant information. In addition, the formation of composite Boolean queries involving the *constituents* elements is not intuitive and the final result for a dataset with dimensionality larger than, say, six is a cluttered display with most of the icons losing their coding due to their reduced size.

[16] A criterion, in this context, means a condition that makes a data item be part of a set in a Venn diagram representation, i.e. a data item d belongs to a set T ($d \in T$).

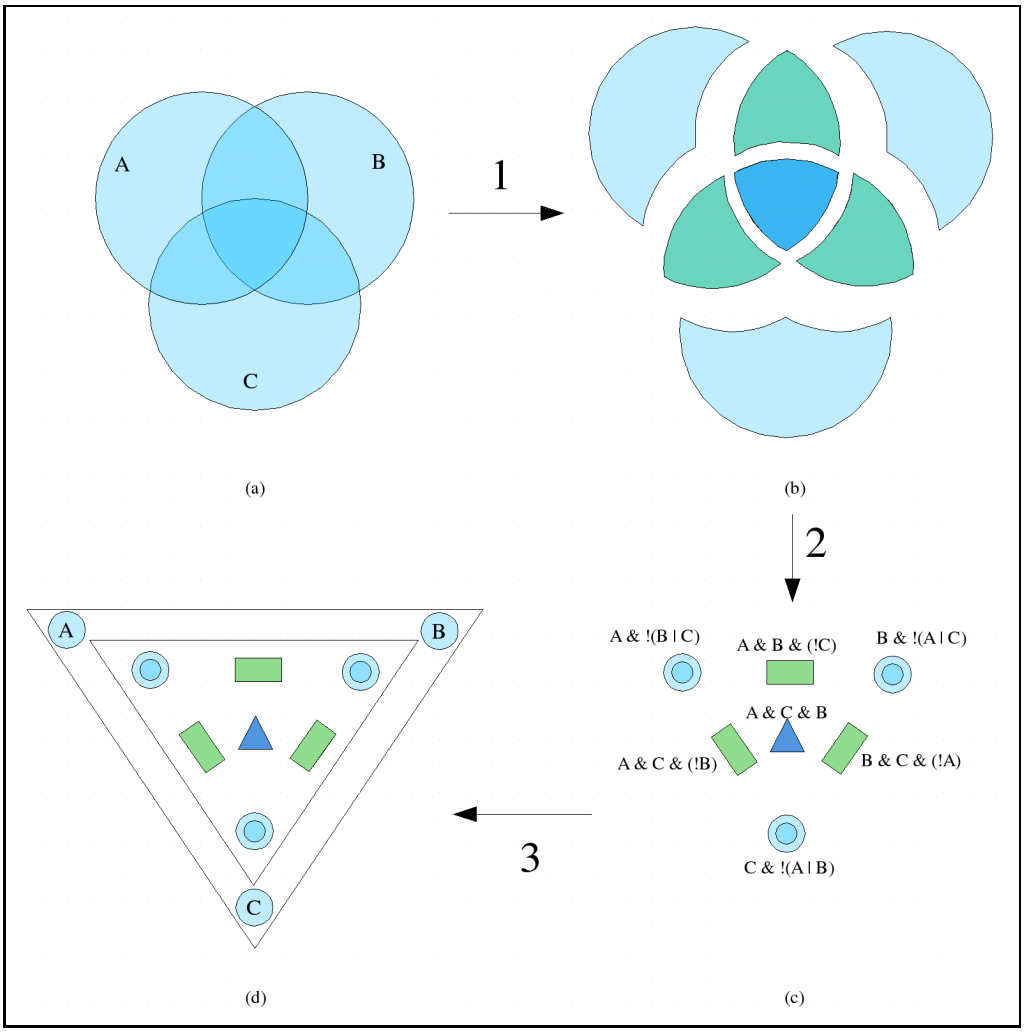


Figure 2.4: Concept of *infocrystal* involving three sets A, B, and C. The Venn diagram in (a) has its components separated in (b) which, in turn, are mapped to icons in (c) and finally organized as an *infocrystal* graph, shown in (d). Note that the original sets are represented in the final picture, on the vertices of the triangle.

2.4.4 Multiple Views Coordination

A view is a visual representation of the data we wish to visualize. When the data is very complex several views of the data may be generated in order to allow users to understand such complexity. The interaction between these multiple views and their behaviour need to be coordinated to enable the user to investigate, explore, or browse the complex data, as well as letting them experiment with different scenarios simultaneously or to compare distinct views.

However, the design and realization of coordinations is usually a difficult and complex task; if the coordination is not well designed this could prevent users achieving their goal;

furthermore, if coordination is well designed but its functionality is not apparent this may increase the cognitive load of the user in understanding the data.

Researchers have recently focused on the issues related to multiple coordination and the results were presented in a conference proceedings [161] and later on compiled in a special issue on coordinated and multiple views in exploratory visualization [162].

North and Shneiderman in [149] have proposed a taxonomy of coordination and their work on this area later evolved into the Snap-Together Visualization [150], which allows users to coordinate visualizations to create multiple-view interfaces that are customized to their needs. The coordination is created and managed according to a conceptual model based on the relational database model. Snap affords several types of coordination such as brushing and linking, drill down, overview and detail, and synchronized scrolling; they can be implemented via a API. Ross and Chalmers in [169] have introduced the HIVE system in which the coordination is defined and controlled using the paradigm of dataflow model and visual programming. Baldonado *et al.* in [204] have introduced a set of guidelines aimed at helping developers to identify when the use of multiple views is appropriate and how to make best use of multiple views when they are employed.

2.5 Summary

The data visualization field may benefit from a more formal abstract structure such as a classification scheme. This mechanism improves the understanding of how techniques are related; clarifies the driving ideas behind methods; can be used to identify how techniques may be improved; and may stimulate the design of new techniques.

Five early proposals of classification schemes and taxonomies were reviewed and their strengths were used to enhance the design of a new proposal for an ontology, based on three stages in the visualization process: *data analysis*, *data picturing*, and *data interaction*. We have found this to be a good strategy to understand the workings of a particular method, as well as being a useful tool to assist separate comparison of techniques in each of the three mentioned stages.

Each method reviewed in this chapter has advantages and disadvantages that are a function of both the type of data we wish to visualize and the visualization goal. As observed by Hinterberg in [84] a technique's visual effectiveness differs between datasets because each technique promotes certain characteristics of the data. Therefore finding the ultimate technique capable of producing effective visualization for all types of high-dimensional data is a difficult task, if not impossible.

Nonetheless, we believe that better methods can be designed if one tries to create them so that both multivariate and multidimensional datasets are treated in a similar fashion. This concept is even more relevant if achieved under a common framework that provides a solid basis for both the systematic design of new methods and a more formal evaluation mechanism.

In the next chapter we shall pursue a common framework whose concept was influenced by the three visualization stage ontology. For that we have adopted the *filtering* sub-category as our first choice of visualization strategy because it has been successfully applied to both scientific and abstract data.

Technique	*References*	*Technique*	*References*
WebBook	[35]	*3D scatterplots*	[14, 117]
3D trees	[163]	*Aggregate Manipulator*	[80]
Andrews curves	[5]	*Animation*	[12, 21, 165]
Bifocal lens	[184]	*Brushing and linking*	[15, 30, 61]
Cam Trees	[163]	*Chernoff faces*	[41]
Circle segments	[8]	*Cluster analysis*	[7, 100, 106]
Graph visualizations	[16, 60]	*Color icons*	[125]
Cone trees	[165]	*Data visualization sliders*	[64]
Details-on-demand	[176]	*Dimension ordering*	[142, 217]
Dimension stacking	[122]	*Document lens*	[164]
Dynamic Query (DQ)	[2]	*FilmFinder*	[1]
Fisheye views	[75, 171]	*GIS*	[63, 121]
Grand tour	[9, 27, 28]	*Heat maps*	[142]
Hierarchical axis	[140, 141]	*Hierarchical parallel coordinates*	[73]
HomeFinder	[210]	*Human project*	[151]
Hyperbolic trees	[120]	*Hyperbox*	[4]
Hyperslice	[199, 200]	*Influence Explorer*	[198]
Infocrystal	[185]	*Information mural*	[102]
Interactive mapping	[20, 29]	*Interpolation*	[154, 173]
Landscapes	[216]	*LifeLines*	[157]
Macromedia Director	[130]	*Magic lens filter*	[187]
M and N plot	[61]	*Mapping categorical data to numerical representation*	[166]
Movable Filters	[187]	*Multidimensional scaling methods (MDS)*	[13, 56, 118, 138]
Multi-line graphs	[48]	*Natural textures*	[98]
Netmaps	[6, 92]	*Networks*	[66]
Neural network algorithms	[114]	*Parallel coordinates*	[97]
Perspective wall	[139]	*Pixel-oriented methods*	[108, 110]
Polar charts	[71]	*Principal component analysis (PCA)*	[99]
Projection pursuit	[55,57,72,96, 103]	*Project management tools*	[186]
Prosections views	[76]	*Quad-tree mapping (QTM)*	[40]
RadViz	[94]	*Rapid serial visual presentation - RSVP*	[182]
Recursive pattern	[107]	*Rotation in the n-dimensional data space*	[55, 90, 148]
Sampling	[51]	*Scalar visualization animation model*	[21]
Scatterplot matrix	[39, 48]	*SeeSoft*	[65]
Selective Dynamic Manipulator	[46]	*Shape coding (autoglyph)*	[17]

Technique	*References*	*Technique*	*References*
SOM maps	[114]	*Spatial display of document collections*	[116, 211]
Spiral	[110]	*Spotfire*	[3]
Spreadsheet-like interface for visualization exploration	[101]	*Star coordinates*	[105]
Starfield visualization	[1]	*Star glyph*	[178]
Stars	[145]	*Statistical procedures: normalization, mean, standard deviation*	[48]
Stick figure (Exvis)'s	[85, 156]	*Survey plots*	[126]
Table lens	[158]	*TreeBrowser*	[87]
Treemap	[175]	*Trees and castles*	[113]
Value bars	[45]	*VisDB*	[110]
Visual Hierarchical Dimension Reduction (VHDR)	[73]	*Visualization for multidimensional function by projections (VMFP)*	[174]
Worlds within worlds	[19, 67]	*Zooming & panning*	[77] [181, Chapter 7]

Table 2.8: Listing the visualization techniques and their respective references in alphabetical order.

Chapter 3

Framework for High-dimensional Visualization

THIS CHAPTER PROVIDES an overview of the research problem and discusses some related issues. From this discussion we identify five challenges, then we elaborate several hypotheses to support a solution to these challenges.

One of these challenges refers to finding a way of treating multivariate and multidimensional data under the same foundation. This has led us to the investigation of previous reference models for visualization, which is explored in the last section. In that we propose a new reference model, extending the dataflow reference model proposed by Haber–McNabb to incorporate the treatment of high-dimensional data.

This reference model together with the TSV ontology described in the previous chapter are the foundations of our proposed common framework for the visualization of high-dimensional data.

3.1 Revisiting the Research Problem

We start by looking at the definitions of the two fields we are trying to bring rather closer together via a common framework. The objective here is to pinpoint mutual elements and shared goals in their definitions.

Scientific visualization (SciVis), according to Earnshaw [25, Chapter 1]:-

> *"Scientific visualization is concerned with exploring data and information in such a way as to gain understanding and insight into the data. (...) The goal of scientific visualization is to promote a deeper level of understanding of the data under investigation and to foster new insight into the underlying process, relying on the human's powerful ability to analyze."*

Information visualization (InfoVis), according to Card *et al.* [32, Chapter 9]:-

> *"Information visualization is the use of computer-supported interactive visual representations of abstract data to amplify cognition. Its purpose is not the pictures themselves, but insight (or rapid information assimilation or monitoring large amounts of data)."*

Looking at these two definitions we promptly identify that both fields are interested in studying effective ways of using the computer as a means of assisting humans with the process of obtaining knowledge about the data (scientific or abstract) under study. Providing insight into data is their shared goal, and both rely on computers to aid achieving that goal. By contemplating the object of study of this research under this shared perspective we may re-introduce part of our research problem as the following question:-

> *How can **visualization** be used to **explore high-dimensional data spaces** and **improve their comprehension**?*

We highlighted in bold the three main elements of this research question. Each one of them corresponds to a core issue of this research:-

1. **visualization**: This element refers to the issue of looking for a suitable visual representation for the high-dimensional entity we wish to explore. According to MacEachren [129] the visualization process is pictured as a cognitive activity in which a human engages (i.e. interacts) to achieve insight about the subject under investigation. This description implies two complementary tasks: *visual representation* and *interaction* (discussed in the next item of this list).

   The task of visually representing high-dimensional data is our first challenge and can be described as **challenge #1**: *To find a visual encoding strategy that could be followed to provide the user with a graphical representation for a high-dimensional entity, which, in turn, would afford the formation of a cognitive map of such an entity.*

2. **explore high-dimensional data spaces**: This element involves both the study of interactive tools and dynamic interfaces, and the numerical multivariate and multidimensional data we are concerned with and want to explore under a common framework.

   Our second challenge could be stated as **challenge #2**: *To find an intuitive and efficient interaction mechanism capable of enabling the user to interactively investigate and navigate through the graphical representation of a high-dimensional data space.*

   Regarding the type of data is **challenge #3:** *To devise a way of tackling in a similar fashion multivariate and multidimensional data, from both scientific and abstract origins.* The idea here is to change the focus from the data to the process of generating a visualization, by introducing abstract levels to describe core elements of the visualization process that work regardless of the type of data or its dimensionality.

3. **improve their comprehension**: This is related to the concept of 'insight' as a mechanism to promote deeper levels of comprehension about the data space. This implies a two step process discussed in the previous chapter – the perception of the data, followed by the construction of a mental model of such data spaces or the improvement of any *a priori* cognitive model.

   This issue also refers to the qualitative aspect of the visualization within the application context, or in other words, the task of ascertaining whether a visualization technique is effective. This evaluation process itself is a complex theme and involves delving into topics such as human computer interaction, perception factors, and qualitative measurement. In fact, according to Grinstein *et al.* [83] and Wong and Bergeron [212], the evaluation and perceptual issues are two of the fundamental problems still facing multidimensional multivariate visualization. So our last challenge is **challenge #4:** *To define and follow a methodology to assess the visualization technique in terms of its effectiveness.*[1]

Next we describe each issue in more detail, considering, during this process, possible solutions for the challenges associated with them.

[1] According to Card *et al.* in [32], "a visual mapping is said to be more effective if it is faster to interpret, can convey more distinctions, or leads to fewer errors than some other mapping."

3.1.1 The Visualization Issue

Finding an answer to the first challenge means solving the complex problem of graphically representing high-dimensional entities on a two-dimensional display. We consider this a complex problem because a dataset's dimensionality is the single most influential factor on the design of a visualization method. When the data to be visualized are defined in an *n*-dimensional space, where *n* is lower than four, there exists a great number of well-known techniques and procedures to deal efficiently with them (see for example Keller and Keller [111]). Indeed, as pointed out by Hibbard in [93] "Visualization has been successful because so much computer data is produced that describe the four-dimensional space-time world that our eyes and brains evolved to see."

For four dimensions one possible strategy is to incorporate the fourth dimension as time and represent the data as an animation of visualizations. But even this solution sometimes does not achieve satisfactory results, especially if we consider, for example, the memory factor – i.e. it is not easy to keep track of all the visualization presented in each frame for comparison, say. The real problem, particularly for multidimensional data, arises when the data dimensionality is greater than four, in which case the range of available solutions starts to lessen.

For multivariate data, however, the dimensionality problem is slightly different. There exists a variety of solutions that can handle more than four variates comfortably (of course, some limitation do exist, perhaps when close to hundreds of variates and thousands of observations). The problem is not so much in terms of the number of variates, but instead it is more related to the question of *how efficiently* a visualization can provide a visual interpretation for the data observations capable of fostering insight into possible relationships among them.

Ideally a representation of high-dimensional data should be designed in such a way as to afford perception by the human mind, accustomed to deal with our four-dimensional space-time world. The effectiveness degree of such visual representation is a function of both the data type and the visualization goal, and, hence, cannot be achieved independently of these factors. Therefore there is no ultimate strategy capable of solving this problem with the same degree of effectiveness for all possible visualization scenarios. For example, according to the intrinsic dimensionality metrics introduced by Grinstein *et al.* in [86], *parallel coordinates* is not as effective as the *RadViz* technique in uniquely identifying data records representing binary vectors, whereas the *RadViz* is not as good as *parallel coordinates* in retaining the original value of individual data observations.

It is feasible, though, to identify several visualization strategies and study how they handle certain types of data and visualization goals. This is exactly what we have tried

to do by enumerating four strategies – filtering, mapping, embedding, and projection – as sub-categories of the *data picturing* stage in the TSV ontology (c.f. Chapter 2, Section 2.3). They have been identified as representative strategies based on the various visualization techniques designed for multivariate multidimensional data.

The *Filtering* strategy

From the four strategies described in Section 2.3 we are especially interested in the *filtering* approach. It comprises those techniques whose central idea is the reduction of the amount of data presented. The filtering process starts by defining a *focus point* in the *n*-dimensional *data space* of interest. The focus point defines the position where the slices are extracted from, during the filtering process.

Usually a slice is low-dimensional, i.e. one-, two-, three-, or even four-dimensional (in which case the use of animation may be necessary), because using higher dimensional slices would lead us back to the original problem of visualized a high-dimensional space. Normally 'thick' slices are utilized to filter multivariate datasets because the *data space* is commonly scattered and sparse, thus the thicker the slice the more data observations are 'filtered'; whereas in the multidimensional case a 'thin' slice is more appropriated since the continuous nature of such ('dense') space allows us to sample it virtually everywhere.

Figure 3.1 shows the filtering being applied to a 3D multivariate dataset to extract a 2D slice defined by the variates X and Z. Note in that picture that only those observations that lie within the slice appear in the final projection shown in Figure 3.1-(d). So if we think of a multivariate dataset as a table – in which data items are rows and variates are columns – the selection of a slice is akin to the creation of a derived table using all rows of the original table but extracting only their values in the selected columns.

For the multidimensional case a slice also corresponds to the subspace spanned by the dimensions selected in the filtering process. In this case, however, a 'thin' slice is used, which means assigning a single value to each unselected dimension, rather than a sub-range as in the multivariate case shown above. Therefore the unselected dimensions are fixed to the corresponding coordinates of a focus point (thus defining the slice's 'thickness'), whereas the selected dimensions are allowed to vary within a specified region (thus defining the slice's 'size').

Figure 3.2 demonstrates the filtering concept applied to a 3D multidimensional space defined by a three-dimensional unit cube, having one vertex at the origin of a Cartesian system and its faces parallel to the canonical planes. In order to 'filter' a two-dimensional subspace of this unit cube the first step is to define the location of the focus point which is used to identify the projection plane (in this case the slice *is* also the projection plane).

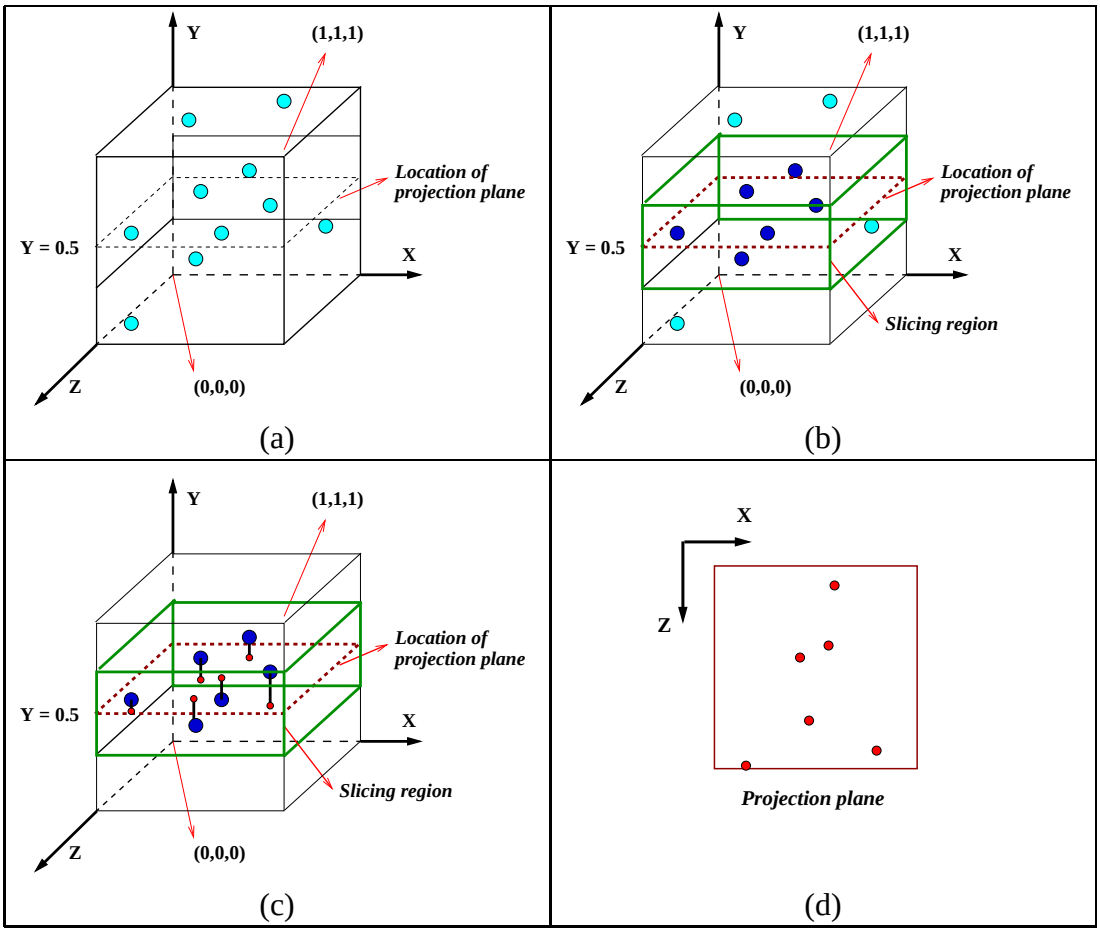


Figure 3.1: 'Filtering' a two-dimensional subspace defined by variates X and Z from a three-dimensional multivariate data defined within a unit cube. The filtering takes place after defining: 1) the position of the projection plane (in this case identified by the coordinate Y of the focus point); 2) the size of the slice (determined by the ranges in X and Z); and, 3) the 'thickness' of the slice to be extracted (determined by the sub-ranges in Y). Picture (a) shows the original multivariate dataset with 10 observations (the cyan balls); picture (b) shows the definition of the slice (in green), highlighting the selected observations in dark blue; picture (c) demonstrates how the 'sliced' observations are projected onto the projection plane; and, picture (d) shows the end result, the projection plane with the 'sliced' observations being represented by red dots.

Then we proceed to select the dimensions to compose the filtered subspace, say dimensions X and Z. These indeed define a two-dimensional subspace (or slice) in the cube, but they alone are not enough to determine the location of the slice within the cube. Finally the location of the 'thin' slice is determined by the current value of the focus point's coordinate corresponding to the unselected dimension, i.e. Y. Now the slice can be uniquely located within the original multidimensional space.

This form of filtering is, somewhat, similar to the philosophy of "divide to conquer", or better put "slice to understand". Another form of controlling the filtering outcome

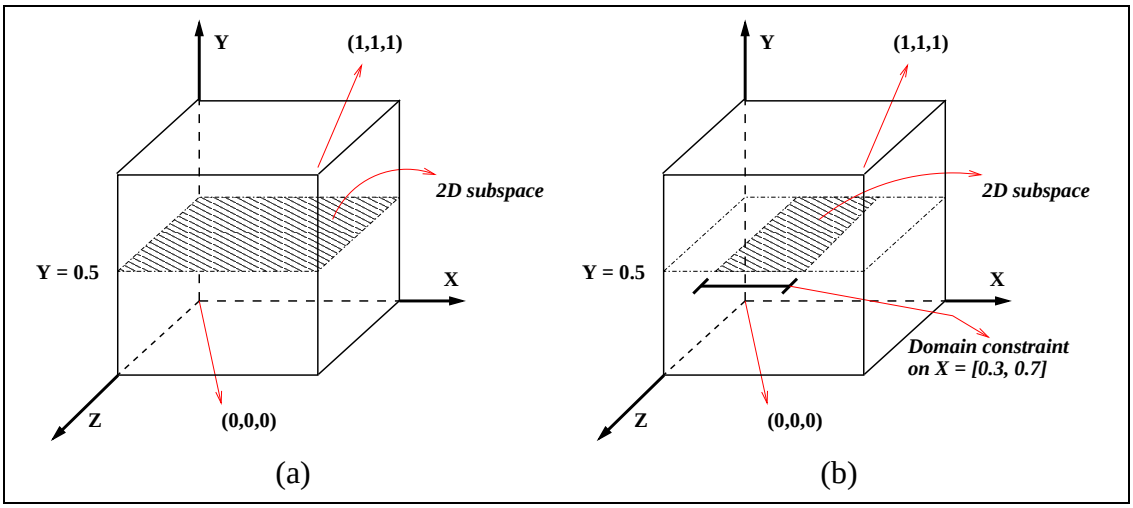


Figure 3.2: 'Filtering' a two-dimensional subspace defined by the dimensions X and Z from a three-dimensional multidimensional unit cube. Besides selecting the dimensions for filtering (X and Z), it is necessary to specify a value for the other dimension ($Y = 0.5$), so that the two-dimensional subspace can be uniquely identified – shown in picture (a). In picture (b) a domain-filtering has been applied to the dimension X, further reducing the size this dimension. Only the data items that lie within that pattern-filled plane compose the 'filtered' data.

in both multivariate and multidimensional cases is achieved when we define constraints on the variates' range (in the multivariate case) or on the dimensions' domain (in the multidimensional range). This type of control has been sown in Figure 3.2-(b) in which the size of the XZ slice has been reduced via a constraint on dimension X.

Normally the control over a dataset's ranges/domains is realized by a simple interface such as sliders associated with each range/domain (see for example the IVEE system [3] for an example of this concept adapted to multivariate applications). This type of filtering is compared to querying a database in search of registers whose values are within some ranges set up (for example via sliders) for each variable. Imposing restriction on a variate's range has the added bonus that the formation of the query (i.e. the selection of the variates to form a slice) is interactive and the result is immediately available on the visualization. It also allows the combination of variates in conjunctions (AND), but disjunctive (OR) combination is also possible if this form of filtering is applied in sequence.

In short, the filtering process may be controlled in two forms:-

1. *By selecting variables from the original* data space *to form a subspace in which the data is to be presented.* The way in which this form of filtering is carried out in both multivariate and multidimensional data can be described by a similar process called '*slicing*'.

2. *By imposing constraints on the ranges for each variate or on the domain of each dimension, thus filtering the data items that will appear on the visualization.* This strategy also works similarly for both multivariate and multidimensional datasets and are responsible for defining the 'size' and 'thickness' of a slice.

3.1.2 The Interaction Issue and Multiple Views

It is well accepted within the visualization field that interaction plays an important role in enhancing the exploratory capabilities of any visualization technique, particularly for those engaged in presenting complex data spaces. This was evident, for example, in the classification taxonomy by Buja *et al.* [29], in which the main focus was on the 'manipulation' part of a visualization technique.

Therefore we believe that for a method to have any chance of overcoming the limitation imposed by the typical low dimensionality of the display devices and successfully provide an effective visualization of high-dimensional data it ought to have not only a good visual representation but also a powerful interaction mechanism. This is so critical that perhaps only a good integration of these components would be capable of delivering an acceptable solution to our main research problem.

Because we are defending the use of the *filtering* as the main strategy for our visualization method, a viable solution for challenge #2 (i.e. finding intuitive mechanisms to navigate a high-dimensional data space) should allow the integration of the filtered subspaces into a multiple-coordinate-views environment, and afford navigation in the *n*-dimensional space to support exploration of the data.

Indeed the use of multiple views brings also other benefits. For example, North and Shneiderman [150] observed that the use of multiple window coordination (which they called *snap-together visualization*) offered an enhanced user performance in data analysis tasks. Also Ross and Chalmers [169] observed that when the coordination allows changes made in one view to be reflected on the others, "interaction can be said to flow between them", which, in turn, permits the user see the focused data of one view within the context of the other. This is what happens, for example, when *brushing & linking* is applied to a *scatterplot matrix*.

3.1.3 The Insight Issue

Visualization methods rely strongly on visual representation to foster insight. Traditionally it is possible to visually represent data using one-, two-, or three-dimensions, but all representations are eventually projected on a two-dimensional display.

A three-dimensional graphical representation is the best choice for presenting data in a visualization when it reflects a physical space inherent to the entity being visualized. That is the reason that makes three-dimensions very popular for SciVis applications, which are primarily concerned with scientific data from our inherently three-dimensional world; and not as popular with InfoVis applications, where usually there is no intrinsic three-dimensional space associated with the entities common in that field.

There is an ongoing debate in the visualization community on the use of 3D *versus* 2D representation. Table 3.1 summarizes arguments for both sides, presented by Card *et al.* [32] and Chalmers [38].

Type of argument	**2D**		**3D**	
	pros	*cons*	*pros*	*cons*
Technological	Users are familiar with paper-style presentation; faster rendering.	None.	Rendering in 3D is no longer a problem.	Interactive real-time high-quality rendering is performance costly; 3D interaction interfaces more complex than 2D ones.
Perceptual	Simplest representation; occlusion is not a problem.	Movement is not restricted to 2D.	3D pointing devices enable smooth navigation without disorientation; additional spatial dimension may encode more information; workspace is expanded due to the depth representation.	Occlusion may be a problem; 3D is better for spatial navigation only if space/model is familiar; representation of text may be a problem; we actually perceive surfaces not volumes.

Table 3.1: Summary of arguments used in the debate on two-dimensional *versus* three-dimensional visual representation of data. The arguments are organized under either a technological or perceptual perspective.

Instead of getting involved in this debate and argue in favour of either of these sides we decided to seek a conciliatory way and take advantage of the positive aspects of both representations. This decision happened naturally as we examined the *difficulty in interpreting 3D visual representations rather than simpler ones*. This was one of the main barriers that van Wijk and van Liere faced when designing the *hyperslice* technique (c.f. Chapter 2, Section 2.4.2.4), and it is related to challenge #1 (i.e. to find a visual encoding to foster insight into the high-dimensional space).

We decided to address this problem with a combination of two measures, which blend naturally with the *filtering* strategy:-

- Allow the user to control the dimensionality of the representation by granting them control over the dimensionality of the subspace as a result of the *filtering* of variables. The user may also interactively expand a subspace's dimensionality, starting, for example, with a one-dimensional subspace and expanding it up to a three-dimensional subspace; or even an animation of a three-dimensional subspace (in this case we may think of it as a four-dimensional subspace in which the fourth dimension is mapped to time in the animation). The user may perform this task in reverse order and return to a lower-dimensional subspace at will.

  We believe this to be a helpful tool in aiding the user not to lose context in the *n*-dimensional space because this progression only adds a new variable to a subspace that they may be already familiar with. Also, because this transformation is reversible the user can always go back to a familiar lower dimensional subspace and experiment with a new combination of variables.

  Banchoff in [11] comments on this idea and suggests that the use of insight obtained in one dimension to understand the next is a good strategy. He defends this idea by affirming (Chapter 1, page 7):-

  > *"We use this process automatically as we walk around an object or structure, accumulating sequences of two-dimensional visual images on our retinas from which we infer properties of the three-dimensional objects causing the images. Thinking about different dimensions can make us much more conscious of what it means to see an object, not just as a sequence of images but rather as a form, an ideal object in the mind.*"

  He concludes saying that the use of several low-dimensional 'images' is a valuable exploration tool that may help us in understanding objects that cannot be placed in an ordinary space.

- Employ standard one-, two-, or three-dimensional visualization techniques – such as *line graphs, scatterplots, isosurfacing* and *volume rendering* – to portray the low-dimensional subspaces instead of creating entirely new visual representations. This simple measure has two side-effects: it avoids the trouble of devising an entirely new visual representation which is likely to increase the cognitive load on the user; and, it grants the user freedom to choose those techniques that they are accustomed to. Hibbard in [93] stressed the importance of the user being able to select interactively and combine different techniques.

Challenge #4 refers to the insight issue and the need for the application of a formal evaluation strategy for a visualization method. Indeed, as noted by Grinstein *et al.* [83] and Wong and Bergeron [212], there still exist many fundamental problems facing high-dimensional visualization. The former stresses that the research effort should shift away from the design of yet more visual displays towards a more rigorous evaluation of experimental visualization techniques. The latter reinforces that the evaluation issue is one of the three cornerstones for further research, together with the *geometric* issues (the data and its representation), and *perceptual* issues (the human and its capabilities). Therefore we concentrate on this important challenge later, in Chapter 7, where we describe our attempt in assessing our proposed visualization system.

3.1.4 Designing a Possible Solution

In this section we assemble several hypotheses as a result of the discussions carried out in the previous sections. These hypotheses were all put together to guide the design of our novel visualization technique for high-dimensional data, called *HyperCell.* It is a visualization technique based on the *filtering* strategy, hence our first hypothesis is: ***The* filtering *philosophy is an adequate strategy to tackle the main research problem***. A comprehensive description of this visualization technique is presented in the next two chapters.

The basic idea behind *HyperCell* is to represent a high-dimensional entity as a group of dynamic subspaces extracted from a specified location in the *n*-dimensional space. Each subspace of the entity contains a graphical representation generated by any standard visualization methods for low dimensions. The subspaces are grouped and coordinated in workspaces, which reflect a specific position within the *n*-dimensional space. Translating this location alters all subspaces within a workspace. Our second hypothesis, therefore, is: ***The use of workspaces to manage the filtered subspaces supports the process of exploring a high-dimensional space.*** For more on using workspaces to manage complex information spaces see, for example, *InfoSpace* proposed by Leftwich [123], *Spaces of Practice* put forward by Büscher *et al.* [31], or *Rooms* by Henderson and Card [104].

We also believe that this simple procedure reduces the overall complexity of the problem, which leads to the next hypothesis: ***Providing a visualization of high-dimensional data via a set of low-dimensional subspaces dynamically created by the user reduces the overall complexity of the problem.***

The control over the creation of subspaces is achieved by means of a two-dimensional user interface, called an *Interaction Graph*. This tool makes use of a full connected graph

in which a vertex represents a variable, and the edges connecting the vertices represent all possible combinations of variables a user may generate. Thus this tool performs the first type of filtering, which is the selection of variables (as described in the item 1 of Section 3.1.1). The same design principle used for the *Interaction Graph* was applied to a second tool, called *n-dimensional Window*. This tool is intended to support the second type of filtering – the filtering of data items by imposing ranges on variables (c.f. item 2 of Section 3.1.1).

We have attempted to keep a consistent representation in both tools by using a similar visual interface for them. They are explained in more detail in the next chapter, and we briefly mentioned them here just to introduce our next hypothesis: ***Our two-dimensional tools portray the high-dimensional space in an intuitive way that supports and encourages the application of a* filtering *strategy to explore it through the creation of several low-dimensional subspaces.***

The last assumption involves the challenge #3, which refers to finding a common way of tackling both multivariate and multidimensional data: ***The use of a reference model based on an extension of the dataflow model by Haber and McNabb can serve as a basis for the understanding of high-dimensional techniques and provide a framework for their implementation in a modular visualization environment.*** This hypothesis is the first to be further explored, in the next section.

3.2 Reference Model for High-dimensional Visualization

Models are important instruments to stimulate the understanding of all the key elements of a visualization process: the visualization core components and how they interface with each other, the data flow, how users interact with the whole process, and even the way the final result is displayed. Different models exist to independently describe all these levels of a visualization and can be used to study the architecture of existing systems or evaluate new ones.

According to Robertson [167] a reference model when formally defined can be used to separate the components of the visualization process by identifying core functionalities. It also can be used as a basis for standardizing terminology, comparison of systems, and identification of constraints or limitations in our understanding of the process.

That is why we decided to look into previous reference models that could describe, at least partially, the process of visualizing high-dimensional data. The motivation is not only to understand the components of such visualization process but also enable the integration of multivariate and multidimensional data into a common foundation.

3.2.1 Reference Models for Visualization

Early work in scientific visualization benefitted from the clarity of thinking in 1990 which underpinned the dataflow reference model of Haber and McNabb [88]. This expressed the visualization process as a sequence of steps: *data enrichment*, to prepare the data for visualization; *mapping*, to convert from numerical data to an abstract geometrical representation; and *rendering*, to create an image from the geometry.

This is illustrated in Figure 3.3. This model has formed the basis for many scientific visualization systems, such as IRIS Explorer [143] and Open Visualization Data Explorer [152], and toolkits such as VTK [172].

The model describes one method of understanding the visualization process and was essentially designed for the core scientific visualization applications, involving scalar, vector field, and tensors, usually defined over 2D and 3D domains.

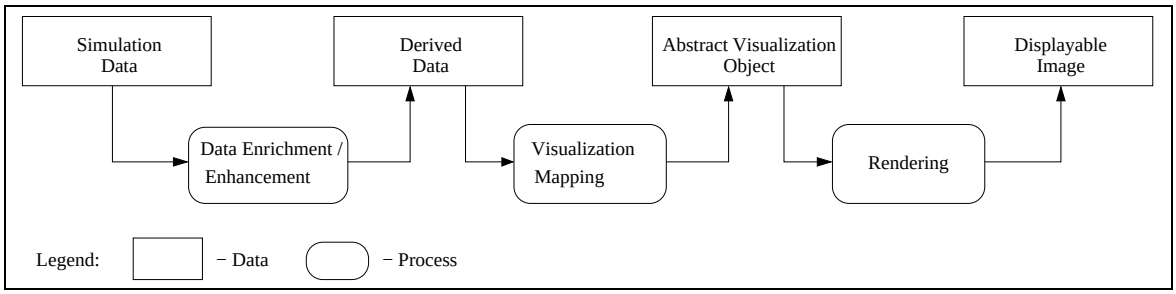


Figure 3.3: Haber–McNabb dataflow model for scientific visualization.

A similar model has also been put forward for the information visualization field, namely the reference model proposed by Card *et al.* [32, Chapter 1]. In that the central element is also the data flow. There are four possible sequential stages for the data – Raw Data, Data Tables, Visual Structures, and Views – and three transformation steps that take the data from one stage into the next – Data Transformation, Visual Mapping, and View Transformation. There is also an extra element, not present in the Haber–McNabb model – the user – who interacts with the model by adjusting the controls in each transformation. Figure 3.4 shows how these stages and transformations interweave.

The model describes several progressive transformations: (1) Data Transformation: takes data stored in any particular format (Raw Data) and converts it into a relational description of data plus meta data (Data Tables); (2) Visual Mappings: takes the Data Tables and maps them to a spatial coordinate system, graphical objects within that spatial system, or even to attributes of those graphical objects (Visual Structures); and, (3) View Transformation: creates the Views of the Visual Structures by specifying parameters such

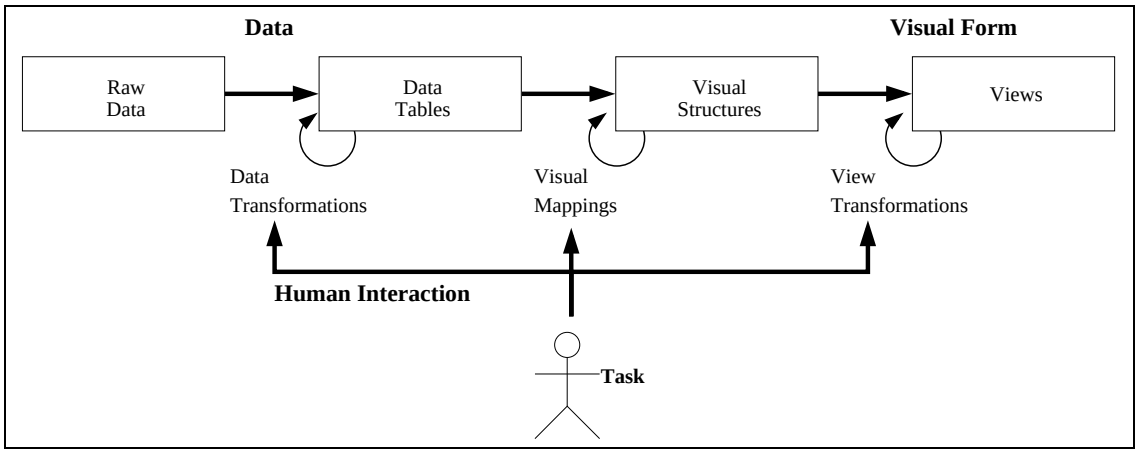


Figure 3.4: Card, Mackinlay and Shneiderman reference model for information visualization.

as position, scaling, clipping, etc. The most important transformation is the Visual Mapping, which is somewhat equivalent to the Visualization Mapping of the Haber–McNabb reference model.

To illustrate the application of this model let us introduce a visualization scenario in which one needs to obtain insight into a list of documents. One possible approach is to use *vector space analysis* (a technique proposed by Salton [170]). In this technique text is interpreted as a vector in a high-dimensional space. The dimensionality of this space is determined by the number of different words used in the collection of text being visualized (not considering common words such as prepositions and articles), thus forming a dictionary. A text is then converted to a vector whose individual coordinates correspond to that text's 'histogram' of all words of the dictionary. It is expected that semantically similar text-vectors would point toward the same general direction within this high-dimensional space. After that it is possible to organize the list of documents (represented by, say, labels) around a circle and connect similar text with lines across the circle. Alternatively one may want to apply a *MDS* algorithm to reduce the dimensionality from the original number of words to a more manageable number, say two or three dimensions. Table 3.2 shows this scenario in terms of Card *et al.*'s reference model.

Chi and Riedl [42, 43] extended this model, creating a reference model that has the same basic structure as the previous models (four data stages and three transformation steps that convert one stage into the next) plus four new classes to account for the different operators that can be used in information visualization techniques. These four classes of operators are applied within each data stage (Value, Analytical Abstraction, Visualization Abstraction, and View) and do not change the underlying data structure of each data stage.

These operators can be *operationally similar* – operators whose implementations are

CARD *et al.* REFERENCE MODEL FOR INFORMATION VISUALIZATION	
Data Stages and Data Transformations	**Visualization Process**
Raw Data	Documents represented as an indexed array of strings.
Data Transformation	*Vector space analysis.*
Data Tables	Documents represented as vectors in n space (n = number of words in the generated dictionary).
Visual Mappings	*Application of MDS algorithm.*
Visual Structures	Documents represented as a table of normalized vectors in 3D.
View Transformation	*Mapping vectors to spatial location in a 3D coordinate space.*
Views	3D scatterplot of documents (proximity in this space means documents semantically similar).

Table 3.2: Visualization of a collection of documents described with Card *et al.*'s reference model (see Figure 3.4). Data Stages are in normal font and Data Transformations are represented in *italic*.

the same across applications – such as rotation, scroll, zoom, browsing a visual representation, changing color maps, etc.; *functionally similar* – operators semantically similar but require different implementations across applications – such as dynamic filtering of value or aggregating data items into clusters; and, *task dependent* – operators especially designed for a particular application domain.

Figure 3.5 depicts the Chi-Riedl model, called Data State Reference Model.

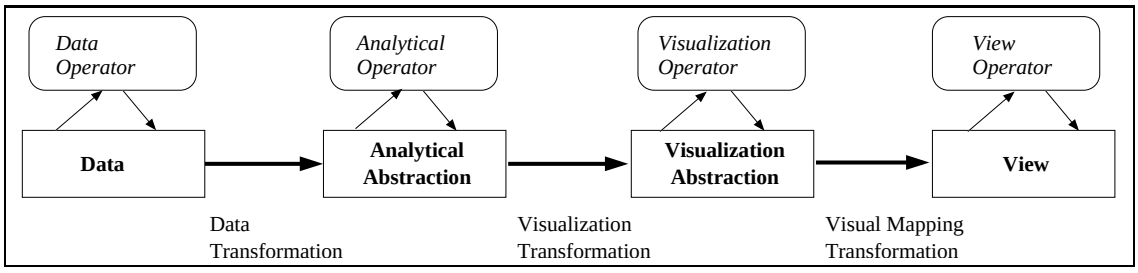


Figure 3.5: Chi-Riedl data state reference model.

Hence the main difference of this model from the others is the introduction of the operator layer that brings an extra level of abstraction. This makes the model more flexible in the sense that the data flow within the visualization process is no longer single-threaded.

At each data stage the flow may diverge, by the application of an operator, to another branch leading to a different visual outcome, and thus revealing other information.

Figure 3.6 presents the same process of visualizing a collection of documents described with the elements of the data state reference model. Note that this figure describes two scenarios. In the left branch MDS is applied and data visualized with a 3D scatterplot. On the right branch a subset is selected after being subject to a Within Stage Operator (*dynamic value-filtering*) and visualized with parallel coordinates.

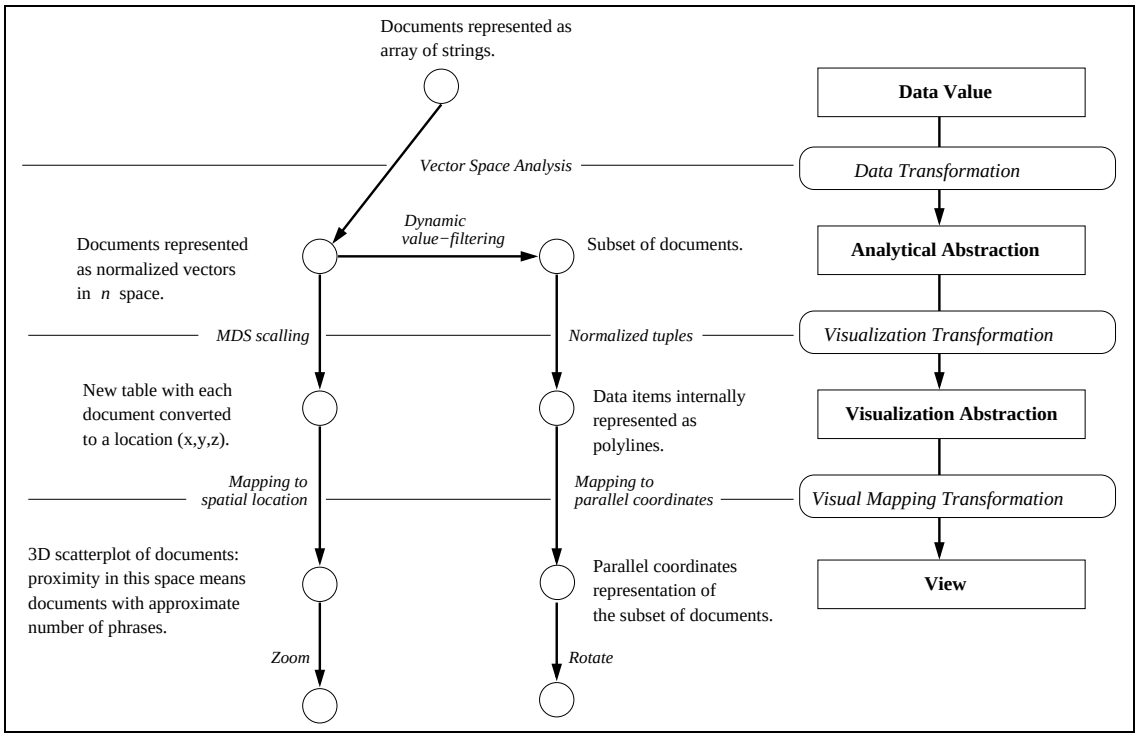


Figure 3.6: The Data State Reference Model (see Figure 3.4) describing the visualization of a collection of documents.

Perhaps the major advantage of this model is the inclusion of the operator layer, which can be used as basis for a modular visualization environment adapted to information visualization.

3.2.2 Extending Haber–McNabb Dataflow Reference Model

In this section we revisit the Haber–McNabb dataflow model: we elaborate the data enrichment step so that it can better describe higher dimensional visualization problems; and then we show how this same model can effectively describe both multivariate and multidimensional problems from information visualization and scientific visualization by using the *filtering* approach.

In this new version there are five data stages (*Problem Data*, *Visualization Data*, *Reduced Data*, *Abstract Visualization Object*, and *Displayable Image*); four data transformation processes (*Data Analysis*, *Data Picturing*, *Visualization Mapping* and *Rendering*); and, one control element (*Data Interaction*). Some of the new elements of the suggested model have been taken from the TSV ontology presented previously in Chapter 2 (c.f. Section 2.3). They are the data transformation process *Data Analysis* and *Data Picturing*, and the control element *Data Interaction*. Figure 3.7 shows all the elements of the model and how the interconnect.

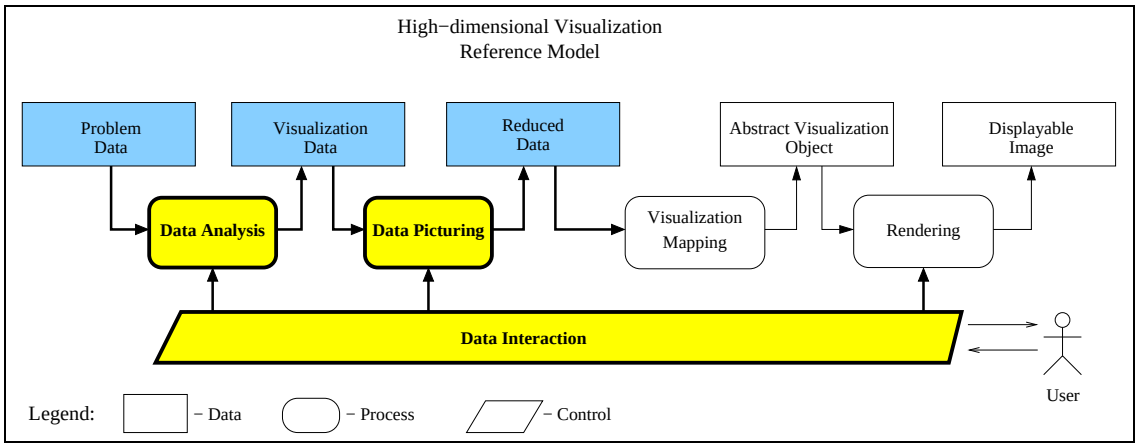


Figure 3.7: Proposed high-dimensional visualization reference model. The new elements of the model are coloured as follows: the categories of the TSV ontology are in yellow, and the three new data stages are in blue. The other modules are the original elements of the Haber–McNabb model (shown in Figure 3.3).

In our extended model, we replace the data enhancement process with two separate processes: 'Data Analysis' and 'Data Picturing'. In the 'Data Analysis' step, the raw data undergo one of the data analysis tasks belonging to the *Data Analysis* Stage of the TSV ontology (c.f Section 2.3.1, Chapter 2). The aim here is to employ a pre-processing procedure to reduce the dimensionality of the data prior to the visualization itself. This could be, for instance, a *multidimensional scaling* algorithm, or a *clustering analysis.* The data analysis step can be seen as a pre-processing step, and it is possible to return to alter the controlling parameters of the task. The manipulation of the parameters that control a data analysis task is described in the model by the control element called 'Data Interaction'. However, changing the control parameters of a data analysis task is the exception rather than the rule. Since there is little interaction with the user, one can see this as a 'computer-centred' operation.

In the 'Data Picturing' step, we apply one of the four strategies described in the *Data Picturing* Stage: *filtering*, *embedding*, *mapping*, or *projection*. For example if the *filter-*

ing approach is adopted this step would involve the extraction of the portion of the data we wish to visualize; while if the *embedding* approach is chosen this would generate an hierarchical structure to accommodate the original data. In other words, the 'Data Picturing' transformation step creates a new data domain that would constitute the basis for the graphical representation to be done in the next steps. In contrast with the 'Data Analysis' step, the 'Data Picturing' step is mostly interactive – the interaction again is accounted for by the 'Data Interaction' control element. The user will typically interact with the system at this level, experimenting with various configurations – thus data picturing step can be seen as 'human-centred'.

Therefore our suggested model works the following way:-

1. We start with our *Problem Data* – this is the original data or it may be the data after a basic transformation whose goal is not to reduce the data dimensionality but simply to prepare the data for the visualization process.

   If we consider the earlier example in which one wishes to visualize a collection of documents this data stage corresponds to the list of documents stored as a set of *n*-dimensional vectors, after being converted from an array of strings via *vector space analysis.*

2. The *Problem Data* might undergo any of the pre-processing methods described in the *Data Analysis* Stage of the TSV ontology, such as *PCA*, *MDS*, or *clustering*, to become the *Visualization Data*.

   In the collection of documents example this corresponds to applying a *MDS* algorithm on the set of normalized vectors in order to map them to a set of triplets (X,Y,Z) corresponding to locations in 3D.

3. The next step takes in the *Visualization Data* and applies any of the data picturing approaches described in the TSV ontology (i.e. *filtering*, *embedding*, *mapping*, or *projection*). The result of the transformation is the *Reduced Data*: the data transformed into a new data domain whose dimensionality is usually lower than the original data and, therefore, can be rendered more easily.

   This is equivalent to representing the documents as spheres in a *3D scatterplot* visualization, using the generated triplets as coordinates for the spheres in the scatterplot.

4. The third and fourth steps correspond to the mapping and rendering processes of the original Haber–McNabb model. The mapping step takes the *Reduced Data*

and creates some geometrical representation, thus generating *Abstract Visualization Object*. The 'Rendering' step creates *Displayable Image* for display on a monitor.

At this stage the spheres representing the documents are created and receive a position in the geometric space of a window where the final image is shown.

The last new element of this model is the presence of the 'Data Interaction' control, which works as a controlling layer between the user (and their task) and the 'Data Analysis', 'Data Picturing', and 'Rendering' processes. The 'Data Interaction' element describes the several types of interaction that the user can apply to control the different transformation processes. This could be, for example, zooming in or out in a *cone tree* representation ('Rendering' level); setting up and controlling a *brushing and linking* element in a *scatterplot matrix* representation ('Data Picturing' level); or, changing the parameters of a *MDS* algorithm to be applied to the data ('Data Analysis' level) or changing the parameters of a interpolant process used, say, in a multidimensional application.

We believe that the introduction of an element in the model representing the interaction activities that may occur in a visualization is an important addition because, as mentioned before, interaction plays an important role in the visualization of high-dimensional data and, as such, deserves a separate treatment. Indeed, as observed by Ma in [128], "a good visualization comes from experimenting with visualization, rendering, and viewing parameters to bring out the most relevant information in the data." Therefore a data visualization system should allow users to explore the parameter space experimentally, using their experience to achieve the visualization goal.

Finally, we think that this new version of a reference model can now better describe a visualization process for high-dimensional data with all its idiosyncrasies, such as the data analysis task commonly used to reduce the data dimensionality. Furthermore this reference model is not restricted to a particular visualization strategy but instead can describe any visualization technique in terms of its three major processes: data analysis, data picturing, and data interaction.

Next we show an instance of this model adapted to describe a novel visualization technique, namely *HyperCell*, that follows the *filtering* approach and can be used to tackle both multidimensional and multivariate data.

3.2.2.1 Filtering visualization for multidimensional data

We begin with the case of multidimensional data, that is, data sampled from a function $F(X)$, where $X = (x_1, x_2, \ldots x_n)$. The visualization mapping and rendering processes are now well understood, but rather less attention has been paid to the data enhancement

process. The original intent was that it should be an interpolation process, for example generating a regular grid of data from a given set of scattered data. In reality it has often been interpreted as a transformation process that selects data of interest from a larger initial set.

In the 'Data Analysis' step of our extended model, the raw data would have associated with it an interpolation function, with the ability to recreate throughout the domain, an estimate of the underlying entity being visualized. One can view this interpolation function being tagged to the data as it passes along the pipeline. It is possible to return to alter the interpolation, but this is the exception rather than the rule.

In the 'Data Picturing' step, we have adopted the *filtering* approach. Therefore at this stage we extract the portion of the data we wish to visualize and generate the 'Reduced Data' (in this particular case it has been called 'Focus Data'). This involves placing bounds on the domain D. We have found it convenient to see this as a pair of distinct operations: the definition of an n-dimensional window with upper and lower bounds, and an n-dimensional focus point within these bounds; together with a constraint term which controls the parameter values within the window – for example, we can reduce the dimension by fixing certain parameters at their focus point values. Thus a slice operation would be seen as both defining a window of interest, and also applying a constraint to specify the slice through the window. The interpolation function created in the data analysis step is used to provide the values of the function on the slice. The filtering process is interactive, thus the user will typically apply a number of filters in a particular session.

The adaptation of the extended reference model to the *filtering* strategy is shown in Figure 3.8. Again we have an overall view as a dataflow pipeline in which one process receives data, operates on it, and passes on the result to another process.

3.2.2.2 Filtering visualization for multivariate data

We now revisit this model from a multivariate data viewpoint. Encouragingly, we find that it describes this case quite effectively. The 'Problem Data' now consists of raw multivariate data $F^i = (f_1^i, f_2^i, \ldots f_k^i), i = 1, 2, \ldots, S$. The 'Data Analysis' step is again computer-centred and consists of some analysis technique. Two popular ones are *Principal Component Analysis, PCA*, which projects the data into a lower-dimensional – i.e. lower number of variates – subspace that accounts for most of the variance in the data [99], and *Multidimensional Scaling, MDS*, which uses nonlinear optimization to lay out the observations in a lower dimensional subspace, in such a way that their separation corresponds as closely as possible to their separation in the original higher dimensional space [138]. Although

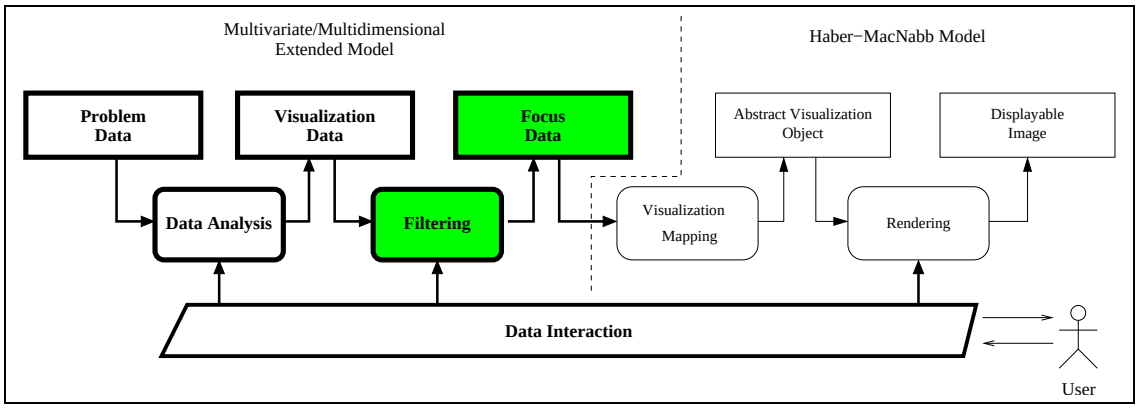


Figure 3.8: The suggested high-dimensional reference model adapted to the *filtering* strategy. The darker blocks on the left-hand side of the dashed line replace the first three components of the original Haber–McNabb dataflow model (see Figure 3.3). The modules in green are the adaptation of the high-dimensional visualization reference model for the *filtering* strategy.

these techniques are not general means for clustering their outcome can sometimes be useful in identifying clusters and trends in the data.

Both *PCA* and *MDS* have the disadvantage, however, that the original set of variates are no longer retained. That is, the data analysis step produces 'Visualization Data' whose variates are not easily interpreted in terms of the variates of the 'Problem Data'. Moreover in extreme cases clusters could be lost by the dimension reduction process. As an alternative approach, aiming to retain the original variates, Yang *et al.* [218] proposed the *Visual Hierarchical Dimension Reduction* (*VHDR*) approach. Here the variates are placed into clusters and a representative variate is selected (either the 'centre' dimension of the cluster, or a new variate which is an average of those in the cluster). This reduces the complexity of the final display, without destroying the meaning of the variates.

The filtering step takes the multivariate 'Visualization Data', however produced, and applies a very similar operation to filtering in the multidimensional case. Again we can see the filter as a pair of operations. We define a window in the value space of the k variates, which we can, as before, interpret geometrically as a k-dimensional region. This specifies the bounds of interest on the values of the variates. In addition we apply constraints, which in this case is a selection from the k variates (similar to the multidimensional case where we used constraints to identify dimensions of interest). In multivariate data visualization, this filtering step of identifying data of interest is often called *brushing*.

The resulting 'Focus Data' (which is the *filtering* 'version' of the 'Reduced Data') then passes to the *Visualization Mapping* step, which applies a suitable technique for multivariate visualization such as those described in Chapter 2 (see for example Section 2.3). Note

that in the case of techniques, such as *scatterplot matrices,* we can see these as requiring (for each scatter plot) a filter which extracts a given two variates (i.e. a slice) from the set of *k*. The final *Rendering* step is as before.

For data which is both multidimensional and multivariate, we can use exactly the same model. The filtering step now applies a filter first to the multidimensional aspect of the data, and then to the multivariate aspect, using the approaches described above. Indeed the filters can be applied in either order. Please refer to Table 3.3 for a summary of how these two operations relate to multidimensional and multivariate data.

Data type	**Data Analysis**	**Data Picturing (Filtering)**
Multidimensional	Interpolation	Window on domain *D*, selection of dimensions
Multivariate	PCA, MDS, VHDR	Window on variate space, selection of variates

Table 3.3: Listing some techniques associated with the Data Analysis & *filtering* steps for multidimensional and multivariate cases.

3.3 Summary

In this chapter re-visited the main research problem, focusing on several related issues. We have introduced a reference model to describe the high-dimensional data visualization process. We have shown that this model relates to the TSV ontology introduced earlier in Chapter 2. These two elements – the TSV ontology and the suggested reference model - comprise the formal basis for a framework that describes the visualization of high-dimensional data under a similar foundation.

We have also described in more detail how the model can be used, for example, to describe a visualization technique based on the *filtering* approach. This particular case is further explored in the next two chapters. Chapter 4 deals with the first step of our proposed visualization technique – *HyperCell,* which involves setting up the filtering parameters and extracting subspaces of a high-dimensional dataset. Chapter 5 deals with the task of organizing the cells into workspaces, in an attempt to build a mental model of the data.

Chapter 4

Implementing the Framework: The *HyperCell* Visualization Technique

---

We introduce a novel visualization technique called *HyperCell* in this chapter. This technique has been designed to address the problem of visualizing multivariate and multidimensional data. Its design is based on the *filtering* approach, which tries to reduce the dimensionality of a dataset (consequently its complexity) by providing tools for the extraction of subspaces – called *cells* – from the original dataset.

Firstly we review some major design requirements that guided the implementation of *HyperCell.* Then we proceed to describe the three core tools that are responsible for the creation of the subspaces needed for the exploration of a high-dimensional data space. Their interface is described and some examples are given to illustrate its functionality. Finally we discuss three enhancements to the filtering process: the incorporation of a fourth dimension, time, in the animation of a 3D cell; the Splitting Cell mechanism; and, the use of *linking and brushing.*

All these tools have been implemented as modules in IRIS Explorer [203]. By doing so we gain access to the data analysis, visualization mapping, and rendering facilities already developed for that environment. Further motivation for this together with implementation details are presented and discussed in Chapter 6. In the next chapter we address the problem of organizing these generated subspaces into a meaningful structure, describing other tools designed for this task.

4.1 *HyperCell*'s Design Guidelines

The major design guidelines considered in the development of this method were listed previously in the Section 3.1.4, Chapter 3, and are summarized below:-

1. *The* rationale *for the HyperCell technique is the* filtering *philosophy, which has been considered an appropriate strategy to tackle the numerical multivariate and multidimensional data.*

   Reasons for that are: (1) the human mind when dealing with complex information prefers to simplify it into small patterns or configurations than trying to grasp it as a whole – hence the advantage of presenting the visualization as a series of low-dimensional 'filtered' subsets; (2) low-dimensional subsets are easier to visualize because there already exist standard well established visualization methods for such cases; and, (3) the filtering approach preserves the original relation between dimensions or variates, as opposed to the other approaches, i.e. mapping, embedding, and projection. (Detailed discussion on this matter was presented in Section 2.3, Chapter 2, and Section 3.1.1, Chapter 3.)

2. *The 'filtered' subspaces are to be organized into workspaces, which are designed in such a way as to represent the region in the n-dimensional space being explored.*

   This type of organization supports the process of exploring a high-dimensional space because a workspace can be thought of as a metaphor for a location in the *n*-space. Therefore changing the parameters that define a window in *n*-dimensional space, such as focus point coordinates and window ranges, automatically affects all subspaces (i.e. cells) stored in the workspace associated with that window.

3. *The visualization of a high-dimensional dataset is realized through a set of dynamic low-dimensional subspaces, which aims to reduce the overall complexity of the task of visualizing a high-dimensional dataset.*

   By low-dimensional subspaces we mean subspaces with up to three spatial dimensions selected from the original set of variables. By dynamic subspaces we mean that the user can, interactively, change a subspace by increasing or decreasing its dimensionality or simply changing the choice of variables that compose a subspace.

4. *Supplying the user with* intuitive *tools with* tight coupling *interfaces [1] enables and encourages the application of the 'filtering' of subspaces.*

   By intuitive we mean that a tool should be designed in such a way as to facilitate the recognition of all available variables plus all possible combinations of them to form

a subspace – this is the essence of the *filtering* process. Tight coupling interface in this context denotes an interface that consistently reflects the current status of the filter environment, which may dynamically change as the user explores the *n*-space.

5. *To develop the elements that comprise the HyperCell technique in such a way that complies with the suggested dataflow reference model for high-dimensional visualization presented in the previous chapter.*

   To achieve that it is necessary to separate the core elements of the *filtering* process into independent modules, which, in turn, are to be implemented in a modular visualization environment.

The 'realization' of the high-dimensional entity we wish to visualize arises from the inspection of several subspaces, grouped into workspaces that reflect location in the *n*-space[1].

4.2 Creating Subspaces: The Filtering Process

To recap, the filter process for multidimensional data defines a window in *n*-space and a focus point within the window, and applies a constraint – in our work here, this constraint identifies those variables which are to be fixed at their values of the focus point and those variables which are allowed to vary within the window.

We have found that this functionality can be achieved using a set of three tools: one defines the window – the *n-dimensional Window*, the second specifies the dimensions – the *Interaction Graph*, and the third extracts the specified data from the original data source (i.e. Visualization Data) outputting results as Focus Data – the *Subsetter*. A schematic of this is shown in Figure 4.1.

Exactly the same interface can be used to filter multivariate data. The *n-dimensional Window* now acts to restrict the range of values of the variates. The *Interaction Graph* selects the variates of interest as in the multidimensional case. Thus we can select a 2D projection for display as a scatter plot, multiple 2D projections for a matrix of scatterplots, or a 3D projection for display as a 3D scatterplot. The *Subsetter* operation then extracts the Focus Data for input to the next stage of the pipeline. Thus *filtering* is applied in a consistent way to both multidimensional and multivariate data.

Before examining each tool that implements the filtering process we shall introduce two running examples that have been used to illustrate their application.

[1] Sometimes we may use ***n*-space** as an abbreviation for ***n*-dimensional space**.

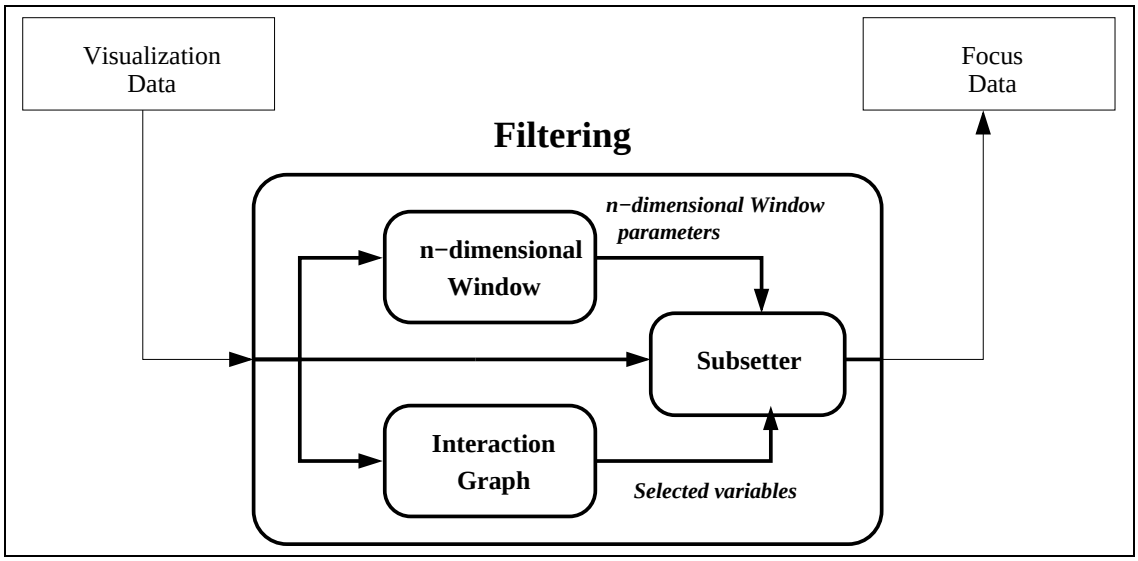


Figure 4.1: Acquiring the Focus Data from the Visualization Data by applying a filtering process. The filtering process is expanded to show its three component operations: window definition (the *n-dimensional Window* module), variable specification (the *Interaction Graph* module), and extracting the corresponding subset of the Visualization Data (the *Subsetter* module).

RUNNING EXAMPLES

The first application is a multidimensional entity: the 4D distance field associated with a four-dimensional line defined by two points within a unit four-dimensional 'cube'. The limits of this four-dimension hypercube are defined as $[0.0, 1.0]$ for all dimensions. For the sake of simplicity the line segment is aligned along a diagonal within the hypercube and its ending points are $P_0 = (1,0,0,0)$ and $P_1 = (0,1,1,1)$, which are two of the sixteen vertices of a 4D hypercube[2]. The data is generated over a regularly spaced grid with 21 points in each dimension and the value at each point is the shortest distance of that point to the line segment.

We are interested in calculating the distance of a point P in 4-space to a point $P(b)$ on a 4D line L, determined by the intersection with the orthogonal line $\overline{P,P(b)}$ dropped from P to L. Therefore the line $\overline{P,P(b)}$ crosses L at b. Figure 4.2 illustrates this geometric interpretation applied to the 2D case for the sake of simplicity, but the formula works for n-D.

[2] See for example [11, page 70] for more information on hypercube in higher dimensions.

Therefore the function is represented as:-

$$f(P) = d(P,L) = \mid P - P(b) \mid = \mid \vec{w} - b\vec{v_L} \mid = \mid \vec{w} - (\vec{w} \cdot \vec{u_L})\vec{u_L} \mid, \quad \textit{where}$$

$$\vec{u_L} = \frac{P_1 - P_0}{|P_1 - P_0|}, \quad \vec{w} = (P - P_0), \quad \vec{v_L} = (P_1 - P_0), \quad P = (x_1, x_2, x_3, x_4), x_i \in [0.0, 1.0].$$

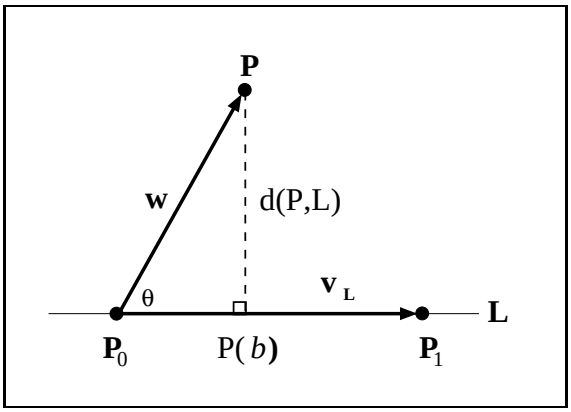


Figure 4.2: Geometric interpretation of a distance d of a point P to a line L defined by two points P_0 and P_1, using a 2D example (*source of image:* [189]).

To demonstrate the *brushing and linking* tool we have chosen to use the Iris flower multivariate dataset, for the application of brushing on this type of dataset seems to be more appealing than the 4D distance field application.

The Iris dataset (introduced earlier in Chapter 2, Section 2.4.2.1) is a four-dimensional multivariate dataset [68], containing 150 observations in three clusters. Each cluster corresponds to one of the three species of Iris flower (*setosa*, *verginica*, and *versicolor*). The dataset is composed of 150 observations (50 of each species, forming a cluster), having four numerical attributes: sepal length, sepal width, petal length, and petal width.

4.3 *n-dimensional Window* – Defining a Region of Interest within *n*-Space

The window definition tool is called an *n-dimensional Window* and we show its user interface in Figure 4.3-(a). From the input data, the *n-dimensional Window* recognizes the number of variables, and lays these out as vertices of an *n*-sided polygon as shown. Each spoke from centre to a vertex acts as a means of specifying the extent of the domain, and the focus point, for that variable.

In the picture on the left, the end-points of the domain are shown as cyan circles (the start of the range) and red circles (the end of the range), and the focus point within that

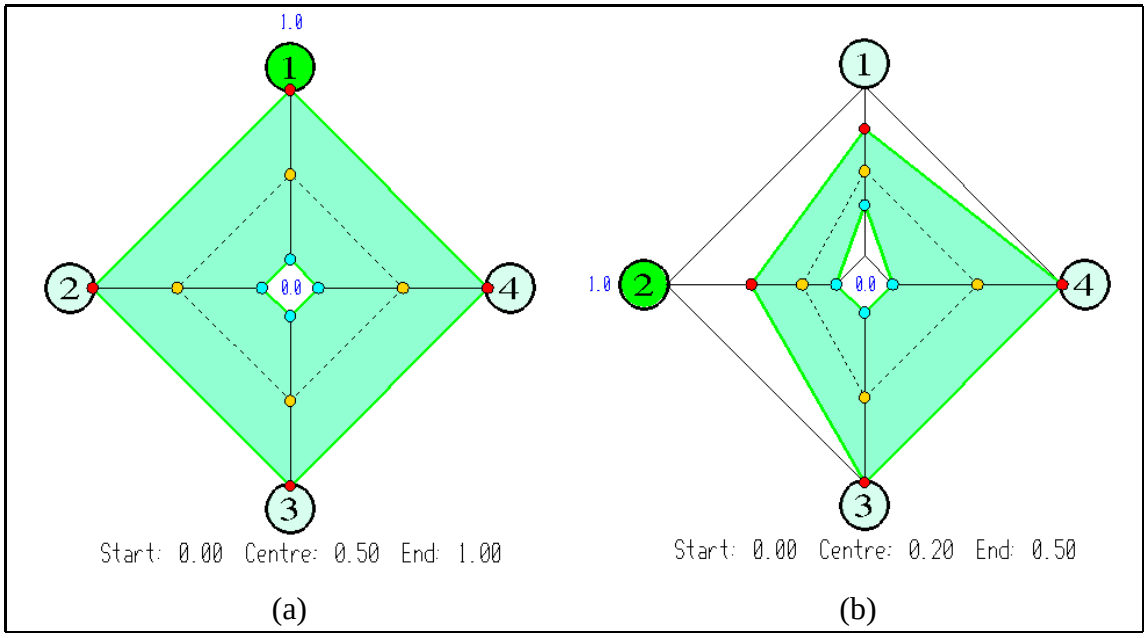


Figure 4.3: User interface for the *n-dimensional Window* definition tool set for the four-dimensional data space of the 4D distance field example – the dimensions lie within the limit $[0,1]$. In picture (a) *dimension-1* is selected and the corresponding minimum and maximum values are shown via a text in blue, respectively, at the centre and just above the selected vertex 1. Note that the current values of the *dimension-1* limits on the diagram are shown at the bottom: *Start* – the cyan control, *Centre* – the yellow control, and *End* – the red control. Picture (b) shows the same diagram after few modifications have been made. The *dimension-1* had its limits changed to $Start = 0.3$ and $End = 0.75$; and, the focus point to $Centre = 0.5$ *Dimension-2* has also been modified and the new values are shown at the bottom because this dimension is currently selected.

domain is marked as yellow. This picture also shows the starting configuration for the tool, having the range for each variable (numbered from 1 to 4) set to cover the whole dataset.

The text at the bottom of the diagram shows three numeric values related to the currently selected variable[3], which is indicated by a saturated green colour. They are: *Start*, the lower limit for the extent of the domain in that variable; *Centre*, the current value for that component of the *n*-dimensional focus point, and *End*, the upper limit for the extent of the domain in that variable. In the 4D distance field example the *n-dimensional Window* starts with all variables set to $Start = 0.0$, $Centre = 0.5$, and $End = 1.0$, thus covering the whole four-dimensional unit hyperbox.

The user can apply different bounds and define different focus points by moving the circles along the spoke using a mouse pointer. Changing these will generate different Focus Data. That is exactly what is shown in Figure 4.3-(b). In that the user has defined

[3] A dimension is selected by a mouse click with the left-most button on any vertex.

subranges on the tool, in particular on variables 1 and 2. Also note that the focus point (represented by a dashed polyline connecting the yellow circles inside the green region) has also been changed to a different position in the four-dimensional space.

4.4 *Interaction Graph* – Selecting the Variables of a Cell

The *Interaction Graph* tool selects the variables to compose a low-dimensional subspace, referred to as a *cell*. The *Interaction Graph*'s user interface is shown in Figure 4.4. Again from the input data, the number of variables are recognized, and these are laid out as vertices of a polygon, maintaining the metaphor of the *n-dimensional Window* definition tool.

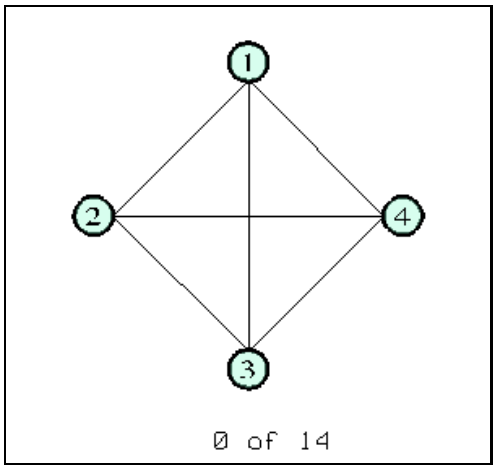


Figure 4.4: User interface of the *Interaction Graph* for four variables. The text at the bottom indicates how many cells have been extracted so far of the total number of possible subspaces.

The overall appearance of the tool resembles that of a 2D fully connected graph in which a single vertex defines a 1D subspace, an edge connecting two vertices defines a 2D subspace, and a triangle based on three vertices corresponds to a 3D subspace. Thus the whole high-dimensional space is concisely represented in this graph diagram.

HyperCell allows the creation of subspaces with up to three variables, therefore it is possible to extract from an *n*-dimensional dataset:-

- n one-dimensional cells.
- $C_2^n = \frac{1}{2}n(n-1)$ two-dimensional cells.
- $C_3^n = \frac{1}{6}n(n^2-3n+2)$ three-dimensional cells.

The total number of possible cells that the user may extract from an *n*-space using *HyperCell* is, therefore, $m = \frac{1}{6}n(n^2+5)$. Presenting all the possible cells simultaneously for a high number of variables is somehow impracticable, given the limited screen space and the large number of three-dimensional cells that would be generated in this process. To avoid this problem we let the user control the process of creating cells to explore the *n*-space. Basically the exploration of an *n*-space, as mentioned earlier on in Section 4.2, may be accomplished either by creating a set of cells (when there is a previous knowledge about interesting slices to look at) or using a *Dynamic Cell*. The use of Dynamic Cells is explained in the next section.

The alternative of transferring control over to the user instead of automatically generating all possible cells has two direct benefits. The first one is that it eliminates the need to have all the data available before the visualization process takes place, delaying it to the moment when the user defines a cell. Only then HyperCell accesses the original source of data and extracts or calculates the piece of ('filtered') data that corresponds to the requested subspace. This is a crucial factor if one considers, for example, the memory requirement necessary to deal with multidimensional datasets. For instance, a simple six-dimensional dataset, having 20 points sampled over each dimension produces a total of 20^6 data points! This issue is even more critical if we think of this process taking place over a network and the data has to be transmitted between nodes of the network to be visualized.

The second benefit is that this approach makes it possible, for example, to integrate *HyperCell* with an evaluation/interpolation module, thus supporting progressive refinement (i.e. the number of data points on the grid is controlled by the user, which, in turn, may avoid altogether the mentioned problem of memory requirement since data is now generated on-the-fly) and possibly computational steering.

Consequently we can affirm that *HyperCell* is a technique that is not constrained, in terms of memory resources, by the dimensionality of a dataset. It only reads into memory those subspaces the user chooses to visualize, rather than requiring the entire dataset to be stored in memory. This also benefits the overall rendering performance because we only need to visualize subspaces with low-dimensionality, which usually can be handled by simple standard visualization techniques (such as isosurfacing, 3D scatterplots, line graph, etc.).

4.4.1 Using the *Interaction Graph* to Manipulate a Dynamic Cell

A Dynamic Cell is designed to assist the exploration of an *n*-space. It implements the search for interesting projections (i.e. interesting combinations of variables) from a given location in *n*-space. This experience is of using a probe to experiment with various distinct orthogonal slices. Once a sufficient number of slices have been tested one may wish to move the probe to another location and repeat the process. When an interesting location is found and representative slices chosen the user may proceed to extract those slices (i.e. the cells) around that spot and store them into a workspace.

An illustration of this probing process should help understand this concept. We represent this in a sequence of pictures – Figure 4.5 and Figure 4.6 – that depict the action of progressively investigating the 4-space via a single dynamic cell. The diagrams on the left represent a sequence of *Interaction Graphs* at different stages of the investigation process applied to the 4D distance field example, and the corresponding data visualizations are shown on the right.

A dimension is selected by mouse-click; in Figure 4.5-(a), *dimension-1* has been selected. This allows parameter 1 to vary within its bounds, while the other parameters, all unselected at present, remain fixed at their focus point values (in this case 0.5). Thus the output will be effectively a 1D line graph, shown on the right hand side of Figure 4.5-(a). That visualization shows that the line segment passes exactly through the centre of *dimension-1*, represented by the bottom of the "V" curve (function value equals zero). Of course, this happens at the the coordinate 0.5 only because the rest of the coordinates are also set at the centre of the hypercube – focus point = $(0.5, 0.5, 0.5, 0.5)$.

A further selection will open the filter to a second parameter, giving a 2D field that can be visualized using a contour map or surface view, as shown in the middle pictures. This visualization, Figure 4.5-(b), shows the orthogonal projection of the distance field associated with the 4D line segment on the plane defined by dimensions 1 and 2. The inclination of the line segment is the same as a diagonal line that runs left to right and top to bottom (depicted by the ellipse-type shape of the colour fields). A third selection will give a 3D field that could be isosurfaced, or volume rendered, as shown in the bottom pictures (isosurface set to 0.3). Notice that lines joining selected vertices in the *Interaction Graph* are thickened and represented in red, and the vertices are highlighted.

The behaviour of the filter has a degree of continuity in the following sense. If we have a 3D field, but toggle off one of the parameters, we create a 2D field which is a slice through the earlier 3D space, at the focus point value of the toggled parameter. Selecting now a fourth parameter, we move into a new 3D space which contains that 2D field as a slice. For instance, we move from the three-dimensional subspace of Figure 4.5-(c), i.e.

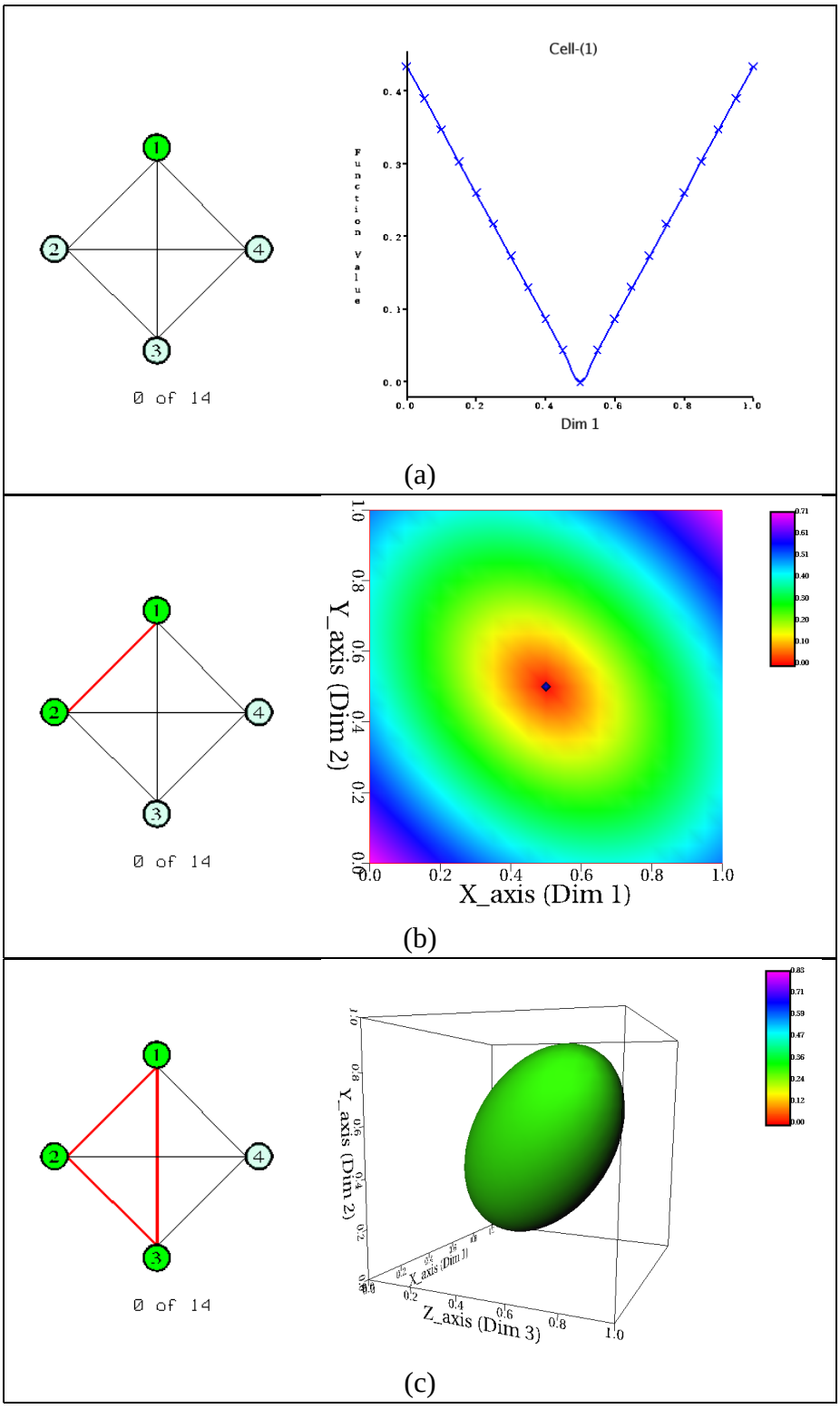


Figure 4.5: User interface of the *Interaction Graph* tool along with several visualizations. This sequence demonstrates the use of a dynamic cell to investigate the 4D space.

cell-(1,2,3), through the two-dimensional slice of Figure 4.6-(a), i.e. *cell-(2,3)*, into the three-dimensional subspace of Figure 4.6-(b), i.e. *cell-(2,3,4)*.

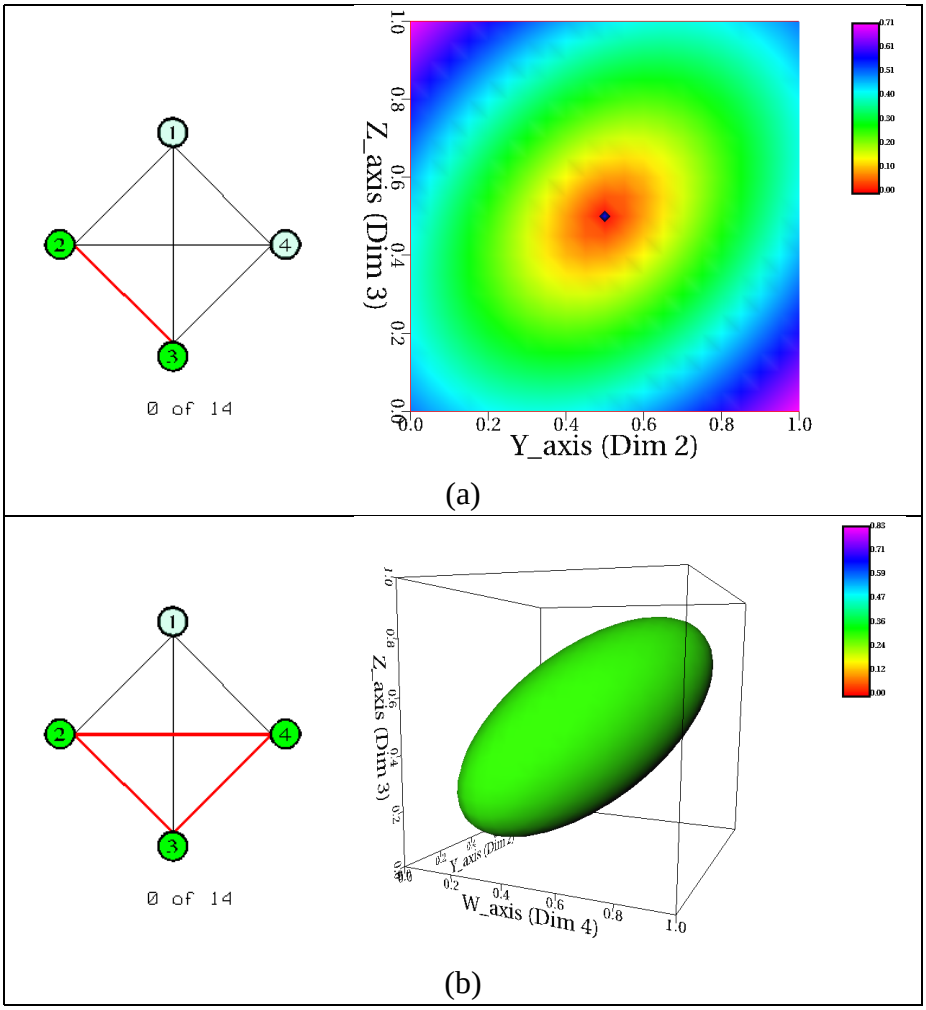


Figure 4.6: Further investigation of the 4D space around the focus point, started in Figure 4.5. Picture (a): *dimension-1* was deselected, generating a 2D coloured field. Picture (b): *dimension-4* was selected, bringing back a 3D visualization – an isosurface (*cell-(2,3,4)*).

4.4.2 Improving *Interaction Graph* with Visual Cues

The functionality of the *Interaction Graph* is available in an IRIS Explorer module called IGraph, whose interface is presented in Figure 4.7. The module incorporates features designed to enhance the power of the *Interaction Graph* mechanism, which are explained in this and the next sections.

The IGraph module keeps a record of all cells that have been extracted with the help of the *Interaction Graph*. This measure avoids, for example, that the user instantiate the

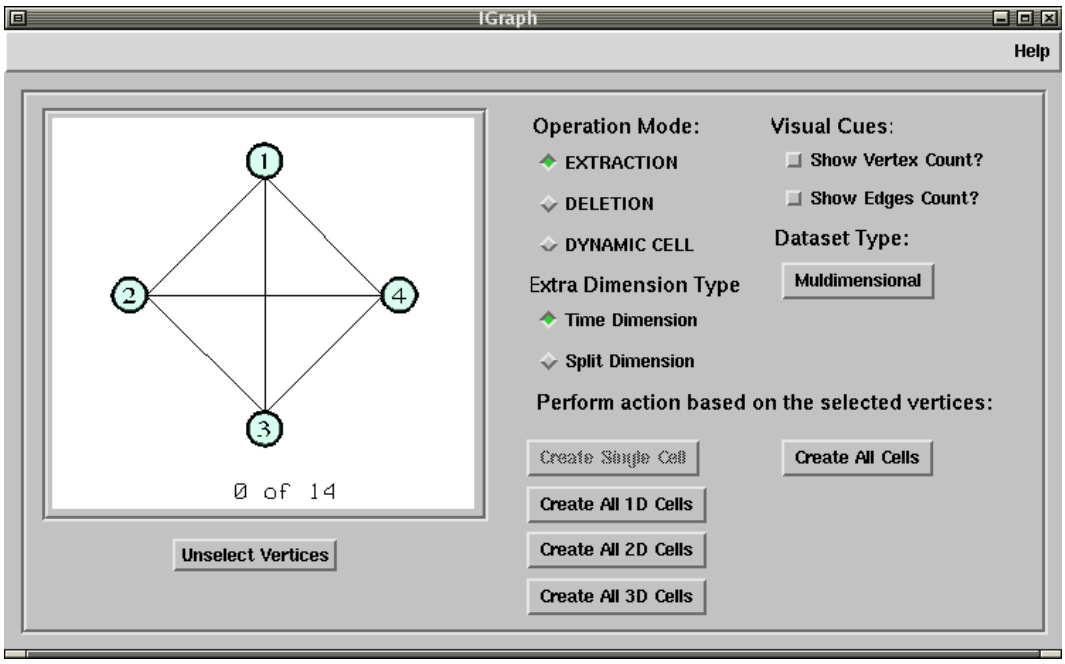


Figure 4.7: The user interface of the IGraph module, which implements the *Interaction Graph* diagram.

same cell more than once. Also it allows the user to 'delete' cells that have already been created. Therefore the IGraph may be also regarded as a manager of subspaces or cells.

The IGraph offers some visual cues to indicate to the user which cells can be created by showing on the diagram how many times a vertex or an edge can still be used to compose a cell. This is activated by two checkboxes located at the top right corner of the module's user interface. Figure 4.8 shows the *Interaction Graph* after the visual cues have been activated. The first picture, Figure 4.8-(a), shows the initial configuration of the diagram in which no cells have been extracted. The numbers located near each vertex indicate how many times a vertex can be used to compose a cell. Likewise the numbers on the middle of the edges indicate how many times a pair of vertices can be used in a cell[4]. Therefore each vertex of Figure 4.8-(a) can be used seven times and the edges can be used three times each to form a cell.

When the *cell-(1,2,3)* is extracted the diagram looks like the one presented in Figure 4.8-(b). Note that now the vertices 1, 2, and 3 can be used only six more times, and the count for the edges 1-2, 2-3, and 3-1 has decreased by one. Notice, however, that there is a 3 over the edge 3-1 but this number actually belongs to the edge 2-4, which is drawn

[4]Because the numbers are located at the middle of every edge sometimes the numbers of different edges will overlap, as it is the case in the example of Figure 4.8 for edges 1-3 and 2-4.

after the 3-1 edge. This problem can be solved by making the correct number associated to an edge to appear on the top if the user 'clicks' on that edge.

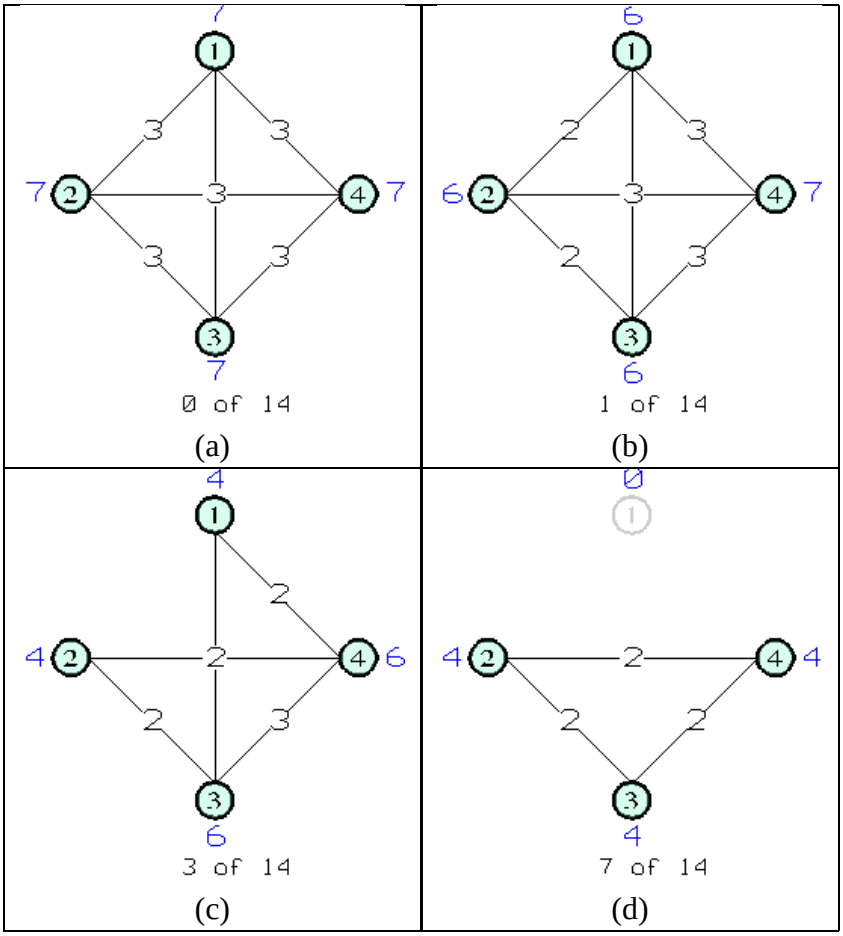


Figure 4.8: Visual cues of the *Interaction Graph*. This series of pictures show the *Interaction Graph* interface through several different stages as the user extracts cells.

When an edge is fully used, i.e. the user has extracted all possible cells that involve that edge, it disappears from the diagram, indicating that it is no longer available to compose a cell. This situation is shown in Figure 4.8-(c), which indicates that the edge 1-2 has been fully utilized, thus the following cells have been created: *cell-(1,2), cell-(1,2,3), cell-(1,2,4).* Finally when a vertex has been fully utilized it cannot be selected and it is represented in a light grey colour with a white background, as shown in Figure 4.8-(d) for vertex 1.

4.4.3 Using the *Interaction Graph* to Create Subspaces

It is important to make clear that the IGraph is not the module that actually extracts the pieces of data corresponding to a subspace from the original dataset. In fact its job is to send downstream the information regarding the cells that the user has requested to another module, called *Subsetter*.

The creation of a subspace is one of the most important tasks, as well as being able to delete any created subspace. Another important task is the probing of an *n*-space using a Dynamic Cell, discussed earlier. The IGraph module implements these three tasks, which can be activated by switching the IGraph to a specific operation mode. This is accomplished by toggling between the three options (EXTRACTION, DELETION, and DYNAMIC CELL) of a radio button on the module's user interface (see Figure 4.7).

In either EXTRACTION or DYNAMIC CELL mode the diagram's interface is exactly as depicted in the previous figures. The only difference is that in the EXTRACTION mode a request for the extraction of a cell is sent only after the user has selected the desired vertices on the diagram and pressed one of the five action buttons located on the bottom right part of the window. In contrast when in the DYNAMIC CELL mode any selection of vertex automatically sends downstream the cell information corresponding to the currently selected vertices, causing the immediate generation of a suitable visualization.

In DELETION mode, however, the interface is slightly different, as shown in Figure 4.9. The *Interaction Graph* in this mode is a 'complementary' version of the *Interaction Graph* in EXTRACTION mode. The difference is indicated by the use of red colouring instead of green. In the DELETION mode the diagram works in the same way as in EXTRACTION mode, the only difference is that the diagram indicates the number of cells that have already been created, rather than the ones that may be created as in EXTRACTION mode. The user selects the cells to be deleted in the same way, clicking on vertices to highlight the desired cell, and then pressing one of the five action buttons located on the bottom right part of the user interface window.

Both processes of 'extracting' and 'deleting' cells are assisted by certain features that obey the tight coupling principle, i.e. giving hints about the current status of the module to guide the user towards the desired result, thus reducing the probability of mistakes being made. For example a vertex may assume five different representations depending on whether it is selected or not and its availability to compose a cell (see Table 4.1).

Another example of tight coupling in the IGraph module is the changing status of the buttons that trigger the extraction (or deletion) of a cell once it has been selected on the diagram. As mentioned before there are five of them, whose labels change appropriately, according to whether the IGraph is in EXTRACTION or DELETION mode:-

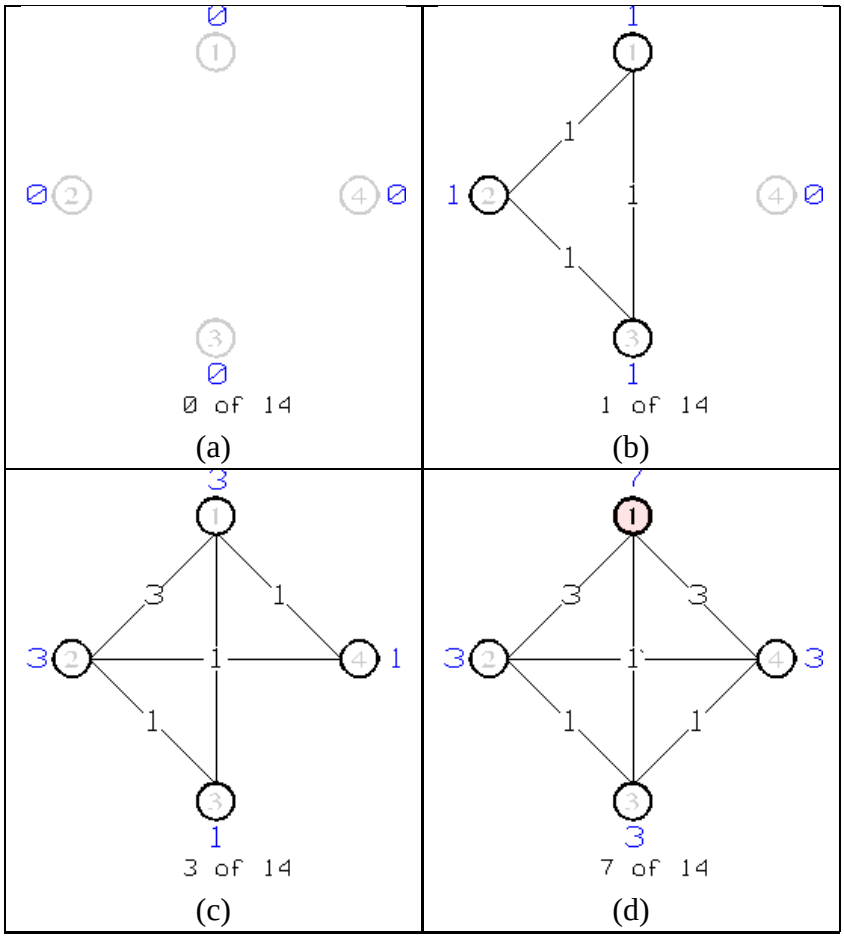


Figure 4.9: This series of pictures corresponds to the same *Interaction Graph*s depicted in the pictures of Figure 4.8, but in DELETION mode.

Vertex Appearance		**Meaning**
Unselected	*Selected*	
		This vertex is available to compose a 1D, 2D, or 3D cell.
		This vertex cannot be used alone in a 1D cell but can be part of other cell(s).
		This vertex cannot be used anymore, individually or as part of a cell.

Table 4.1: List of possible visual representation of a vertex's status in the *Interaction Graph*. The same representation is true for the *Interaction Graph* in DELETION mode, except that the red hue is used instead of green.

- **Create/Delete Single Cell**: Extracts/Deletes the currently selected cell on the diagram. This button becomes active *only* if a valid cell of any dimensionality is currently selected on the diagram.

- **Create/Delete All 1D Cells**: This button is active *only* if no vertex has been selected on the diagram *and* there is at least one 1D cell to be extracted. If this button is pressed a request for the extraction/deletion of all possible 1D cells (not yet created) will be sent forward.

- **Create/Delete All 2D Cells**: This button is active *only* if zero or just one vertex has been selected on the diagram. If no vertex is selected pressing the button will request the extraction/deletion of all possible 2D cells not yet processed. Pressing the button while one vertex is selected will request the extraction/deletion of all possible 2D cells not yet processed that *contain* the selected vertex. Therefore if only one vertex is selected and there are no 2D cells containing the selected vertex available for either operations then the button is disabled.

- **Create/Delete All 3D Cells**: This button is active *only* if zero, one, or two vertices have been selected on the diagram. If no vertex is selected pressing the button will request the extraction/deletion of all possible 3D cells not yet processed. Pressing the button while one vertex is selected will request the extraction/deletion of all possible 3D cells not yet processed that *contain* the selected vertex. The same happens if the button is pressed while two vertices are selected: all possible 3D cells involving that pair of dimensions will be extracted/deleted, unless they have already been processed. Again if one vertex or a pair of vertices are selected and there is no 3D cell involving the selected subspace available for either operations then the button is disabled.

- **Create/Delete All Cells**: This button is active while there is at least one cell of any dimensionality available for extraction/deletion. It creates all possible cells starting from the selected vertex or vertices. For example, if the vertex 1 is selected and this button is pressed then it will request the extraction/deletion of the 1D cell involving variable 1 and all possible 2D and 3D cells that contain variable 1 but have not yet been processed. If this button is pressed while no vertex is selected it will request the extraction/deletion of all possible 1D, 2D, and 3D cells that have not been processed so far.

Table 4.2 shows a summary of the activation of these buttons depending on the number of vertices selected on the diagram.

Button	Number of vertices selected			
	0	1	2	3
Create/Delete Single Cell	–	A	A	A
Create/Delete All 1D Cells	A	–	–	–
Create/Delete All 2D Cells	A	A	–	–
Create/Delete All 3D Cells	A	A	A	–
Create/Delete All Cells	A	A	A	A

A: Button active (enabled).

Table 4.2: Listing the status of the buttons that trigger the extraction/deletion operations on the IGraph module's interface. Their availability is affected by the number of vertices currently selected on the *Interaction Graph* diagram.

4.5 *Subsetter* – Extracting the Cells

The final module of the diagram that describes the filtering process (presented in Figure 4.1) is the *Subsetter*. It is responsible for extracting the Focus Data (i.e. the subspaces) from the original dataset to form the cells. The *Subsetter* receives two inputs: the parameters that define a window in *n* space, sent by the *n-dimensional Window* (i.e. the limits on variables and the focus point); and, the list of cells to be extracted sent by the *Interaction Graph*.

In terms of IRIS Explorer, this module receives lattice data of any dimension as input or simply the dataset file name. Ideally this module should receive only the dataset file name and perform the extraction directly from the file, without reading all the *n*-dimensional lattice into memory. However, the current implementation needs to have the whole lattice in memory (either directly via the pipeline or read in based on the file name provided) in order to successfully extract a subset.

The resulting output is a lattice that may be sent to a any suitable mapping module in IRIS Explorer. There are two output cases, depending whether the input dataset is multidimensional or multivariate. In the first case – multidimensional input – the output is a one-, two-, or three-dimensional lattice as a result of a slicing process. in the second case – multivariate input – the output is a one-dimensional lattice expressed in curvilinear coordinates.

In the multivariate case the number of coordinates per data point is equal to the dimensionality of the selected cell. The other variates – the ones that have not been selected – can optionally be sent in the lattice as the vector components of each data point. Consequently the data from the unselected variates can still be available for each data point

or observation. This is useful because this extra information may be used, for instance, to colour a sphere in a *3D scatterplot* representation.

It is also possible to generate 'animated' data by selecting a *base cell* (up to 3D) and an extra variable as the time dimension. In this case the Subsetter extracts an array of *base cells*, each one of them is extracted as the 'time' variable varies stepwise through its domain values – this is the mechanism used to create 4D cells.

4.6 Enhancing the Filtering Process

In this section we discuss three features aimed at improving the investigation capabilities of the filtering process described so far. These are the use of a fourth dimension as time in an animation of a low-dimensional subspace; the Splitting Cell mechanism; and, the implementation of the *linking and brushing* technique in a module called *n-dimensional Brushing* or simply NDBrush.

4.6.1 The Time Dimension

Whenever the user selects variables using the *Interaction Graph* tool to compose a cell they determine the so-called *free* variables, i.e. the selected variables are 'free' to vary over their range defined in the *n-dimensional Window*; whereas the remaining unselected variables are considered *fixed* variables, for they are associated (i.e. fixed) to the values of the focus point, also defined in the *n-dimensional Window*.

One of the fixed dimensions can be freed and used as time dimension in an animation. In our *Interaction Graph* tool this is done by clicking on one of the unselected vertices with the right-most button of the mouse, which causes a vertex to be highlighted in yellow, as depicted in Figure 4.10.

The 'duration' of the animation is determined by the range of the variable selected as the time dimension. The sequence of time frames in the animation is generated by moving the coordinate of the focus point associated with the time dimension, stepwise along its defined range. At each new coordinate a *base cell* (i.e. the initially selected cell, without the time dimension) is extracted and stored in the output lattice, taking into consideration the updated coordinates of the focus point. Hence, the final output is a multi-step lattice.

Spence [181] observed that animation might be a powerful tool, especially if one observes the value of some property X of an artifact as the value of some property Y is manually or automatically varied. This might help the formation of an internal model of

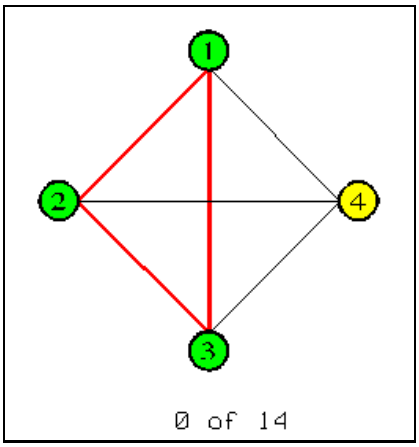


Figure 4.10: This picture shows the *Interaction Graph* in which the *dimension-4* has been selected (vertex in yellow) as time dimension in the animation of *cell-(1,2,3)*.

the relation between X and Y. One possible interpretation may be that there appears to be some trade-off between these two properties.

4.6.2 The Cell Splitting Mechanism

Cell Splitting is an extra feature that we devised to enhance the visualization of a subspace. It consists of 'splitting' one of the variables of a cell to accommodate another variable. An example will illustrate this concept.

Consider a 3D cell, say *cell-(1,2,3).* We may want to split the *variable-1* and merge it with *variable-4* to observe any degree of continuity between them. The result is a *cell-(1:4,2,3),* which is equivalent to visualizing at the same time the cells *cell-(1,2,3)* and *cell-(4,2,3)* (actually *cell-(2,3,4)* because the variables are always sorted to keep consistency in the representation). Figure 4.11-(a) shows a schematic for this process.

The 'splitting' occurs exactly at the coordinate of the focus point. Therefore both dimensions involved in the splitting process must have the same focus point coordinate. Let us suppose that in our example the focus point coordinate is initially located at the centre of the four-dimensional hypercube, $(0.5, 0.5, 0.5, 0.5)$. After the splitting takes place we have a new subspace in which the data is a combination of $f(x_1, x_2, x_3, \hat{x}_4), x_1 \in [0.0, 0.5]$ and $f(\hat{x}_1, x_2, x_3, x_4), x_4 \in [0.5, 1.0]$.

The Splitting Cell mechanism is activated on the user interface of the IGraph module by clicking on a radio button that switches between Time dimension (discussed next) and Split dimension. The first step in selecting a splitting cell is to define the *base cell*, in the example clicking on vertices 1, 2, and 3. Then to indicate which vertex is to be accommodated in the split cell the user clicks on any unselected vertex with the right

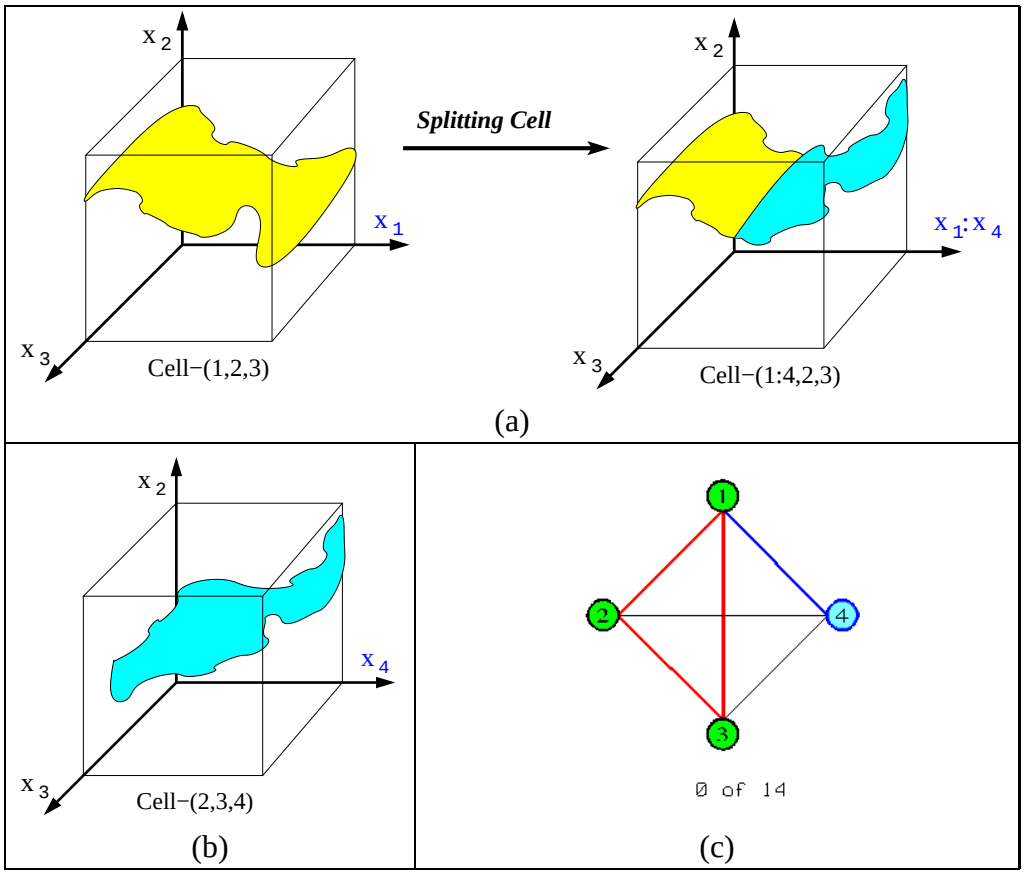


Figure 4.11: Schematic illustrating the Splitting Cell process. Picture (a) shows the Splitting Cell process of *cell-(1,2,3)* into *cell-(1:4,2,3)*, which involves *variable-1* and *variable-4*. Picture (b) shows the original *cell-(2,3,4)* which has been merged with the *cell-(1,2,3)*. Picture (c) shows how the *Interaction Graph* looks when the user selects a 3D subspace, *cell-(1,2,3)*, and applies the Splitting Cell mechanism to include *variable-4*.

most button. In our case there is only one option: vertex 4. This causes the edge joining vertex 4 and vertex 1 to be thickened and represented in blue. To change the dimension of the *base cell* the user needs to click on any of the selected vertices, again pressing the right most button of the mouse. This makes the edge linking these two vertices thicker and blue. Finally clicking on the 'Extract Single Cell' button will generate a split cell.

Note that splitting a cell to accommodate an extra variable is different from creating two cells that share a common variable and putting them side by side. In this case each cell would have its own focus point representation somewhere within the cell limits, whereas in the splitting cell procedure there is only one focus point within the split cell.

Figure 4.12 demonstrates this principle applied to the 4D distance field example. We have chosen *cell-(1,4)* as our *base cell* and have split *dimension-1* to accommodate *dimension-2*, thus creating *cell-(1:2,4)*.

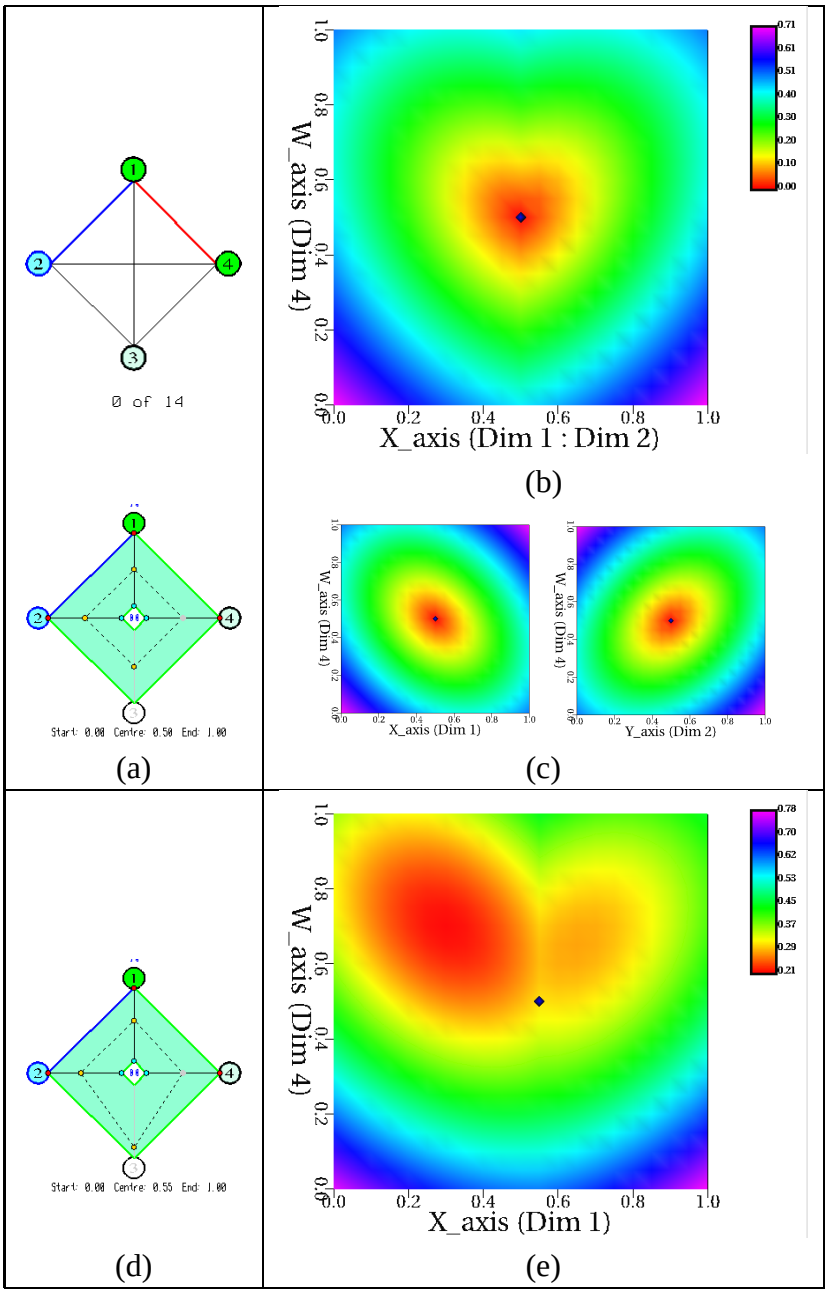


Figure 4.12: Applying the Splitting Cell to the 4D distance field dataset. Picture (a) shows the *Interaction Graph* (top) and the *n-dimensional Window* (bottom) diagrams set for the *cell-(1:2,4)*, shown in Picture (b). Picture (c) shows the original *cell-(1,2)* and *cell-(1,4)* which are merged into the split cell of picture (b). Picture (d) shows the same *Interaction Graph* after the focus point has been moved along the *dimension-3* and *dimension-1:2*, which generates a new visualization for *cell-(1:2,4)* presented in Picture (e). The focus point is represented in each visualization by a blue dot.

In order to facilitate the manipulation of the filtering region via the *n-dimensional Window* and ensure that the coordinates of the focus point are the same for both variables

involved in the splitting process it is possible to feed the output of the IGraph module (i.e. the request for a split cell) into the *n-dimensional Window*. This makes the ND-Win (the module that implements the *n-dimensional Window*) restrict the access to the *n-dimensional Window* diagram, by partially disabling the vertices that are not involved in the cell, as shown at the bottom of Figure 4.12-(a). Note that in that *n-dimensional Window* diagram the vertex 3 is partially disabled and the starting and ending controls that define the limits on that dimension are not active (changing the limits on that does not affect the *cell-(1:2,4)* because this dimension is not in the cell). In addition if the user moves the focus point controls of either *dimension-1* or *dimension-4* the other moves accordingly, ensuring that both have always the same coordinate.

Figure 4.12-(d) shows the *n-dimensional Window* after some alteration has been made to the focus point (on *dimension-3*) and the corresponding visualization of the *cell-(1:2,4)*, in Figure 4.12-(e). This small experiment has shown us that the symmetry between *dimension-1* and *dimension-2*, present in Figure 4.12-(b), has been lost as we move the location of the focus point, as shown in Figure 4.12-(e).

4.6.3 *n-dimensional Brush* – Linking the Cells

The *brushing* technique is implemented through the *n-dimensional Brush*, which is available in *HyperCell* via a module called *NDBrush*. The definition of the brush object obeys a metaphor similar to the *n-dimensional Window*, because the brush object can also be regarded as a window in *n*-space. Therefore the user interface of the *n-dimensional Brush* diagram is based on the *n-dimensional Window*'s, as shown in Figure 4.13. The differences are the use of yellow colour instead of green, and the absence of a focus point.

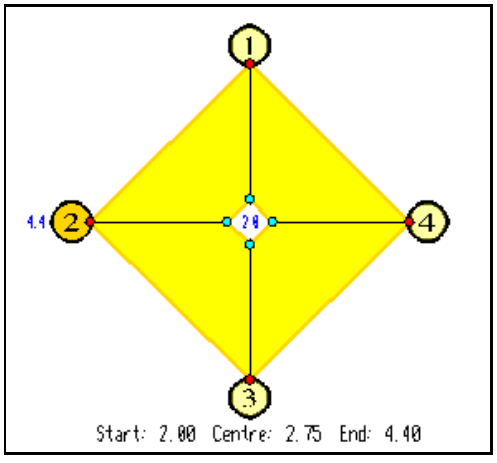


Figure 4.13: User interface for the *n-dimensional Brush* set for a four-dimensional dataset.

We have implemented a simple version of the *brushing* technique in the sense that the output of this module is a bounding box rendered on a cell's visualization window in red colour. Nonetheless, this simple implementation is powerful enough to enhance the exploration of a high-dimensional dataset, especially multivariate ones. This is illustrated in Figure 4.14, which shows a brush object applied on all possible 3D cells of the Iris flower dataset.

One of the three clusters of the Iris dataset has been 'brushed' and the result is reflected in all four 3D cells, depicted by the red wire-frame rectangle. The clusters can be easily identified from the dataset because the observations are ordered, respectively, by the three species of Iris flower: *setosa*, *verginica*, and *versicolor*. So in the particular case of Figure 4.14 we have set the brush to cover the observations 50 to 100, thus isolating the middle species.

The *brushing and linking* mechanisms works in a similar way when applied to multi-dimensional data, acting as a device to identify 'regions' throughout linked views instead of data elements as in the multivariate case.

The 'linking' between subspaces or cells is accomplished through another module that implements a workspace in which the subspaces are organized. This and other issues such as navigation are discussed in the next chapter.

4.7 Summary

In this chapter we have discussed the main elements of our proposed visualization method called *HyperCell.* This visualization technique is the implementation of a filtering approach, and its design follows the suggested framework for high-dimensional data introduced in the previous chapter.

The filtering process is accomplished by means of three core tools: the *n-dimensional Window*, which sets the filter coverage; the *Interaction Graph*, which selects the variables that compose the filtered data; and, the *Subsetter*, which extracts the filtered data from the original source according to the parameters defined by the first two tools. The user interfaces of both *n-dimensional Window* and *Interaction Graph* tools follow a similar metaphor: to represent the variables of the high-dimensional data as vertices in a polygon, and subspaces as lines connecting these vertices.

We have explained how the *Interaction Graph* can be used in conjunction with the concept of a Dynamic Cell to probe the *n*-space, moving smoothly between subspaces with distinct dimensionality. This is thought to be a helpful mechanism because the user may start the probing of the *n*-space through a one-dimensional cell (supposedly easier to

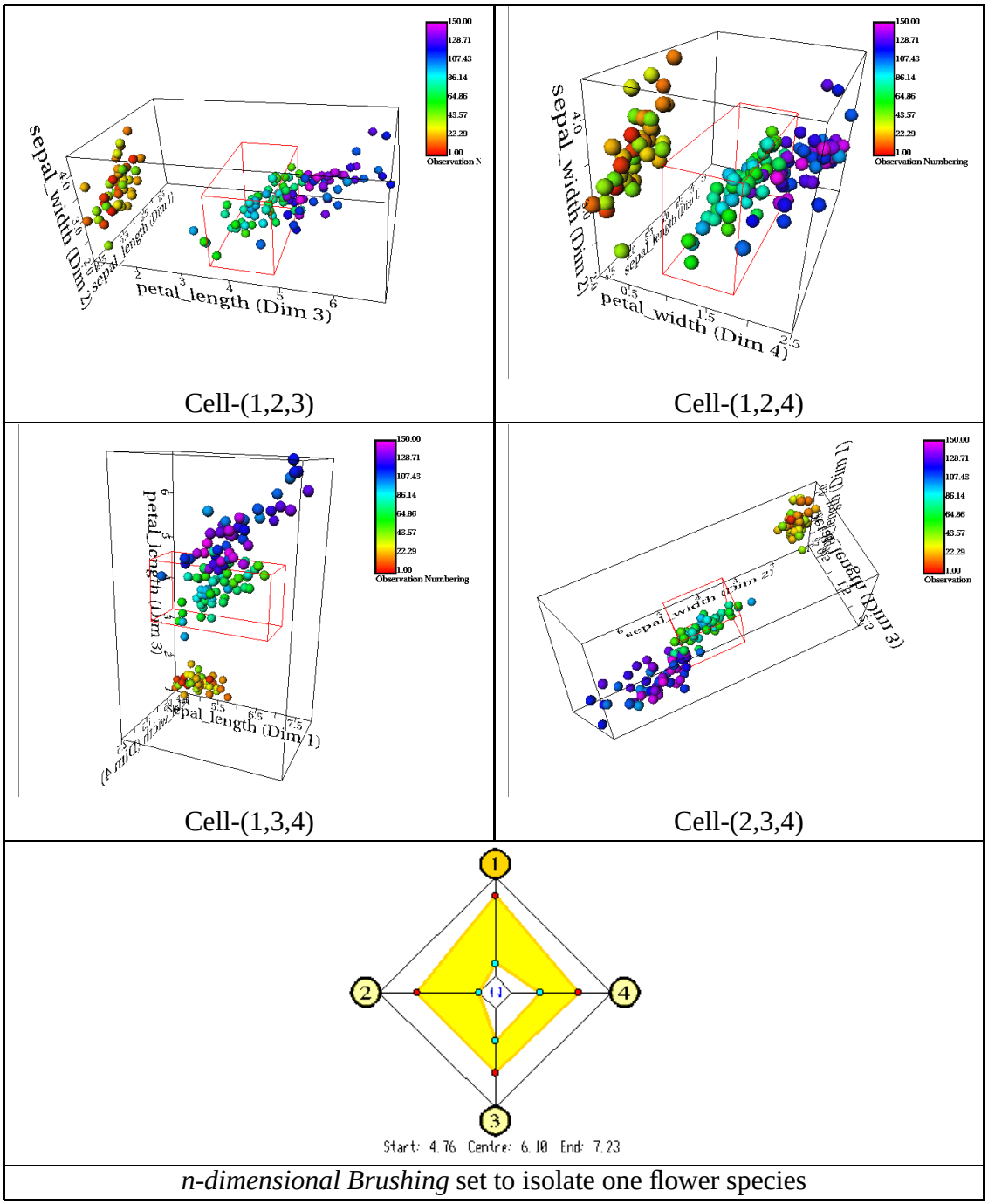


Figure 4.14: These pictures show the use of the *n-dimensional Brushing* on the four-dimensional multivariate Iris dataset. The spheres are coloured by the order of the observations in the dataset corresponding to the different species, and one cluster has been brushed in all four possible 3D cells.

understand), get used to this subspace and then progressively add another variable(s) to this familiar low-dimensional cell. This simple action ensures that the exploratory process

always takes place within a familiar context, and because this operation is reversible the user can go back to the familiar lower dimensional cell and re-start the exploration from there. We have also shown that the filtering process works similarly for both multidimensional and multivariate datasets.

The tools have been implemented as IRIS Explorer modules. In this chapter, however, we have focused only on the concepts behind each tool, saving the implementation details for Chapter 6. Furthermore, we have discussed a few enhancements to the filtering process, namely the mapping of a fourth dimension to time in an animation of a 3D cell, the Splitting Cell mechanism, and the implementation of a *brushing and linking* technique.

Although the filtering process covered in this chapter is considered the major component of the *HyperCell* technique, it is clear that extracting several low-dimensional subspaces is only one stage of the exploration process. We need also to provide the user with some device that helps organize the extracted subspaces, and mechanisms that afford navigation within the *n*-space. These issues are explored in the next chapter, where we present the remaining tools that are part of the *HyperCell* technique and were designed to address these questions.

Chapter 5

Exploring the *n*-Space with *HyperCell*

SO FAR WE have described the main *rationale* behind *HyperCell*: a filtering process that creates several low-dimensional subspaces, called *cells*. We have also explained the tools that implement the filtering process, making them available as a set of IRIS Explorer modules. These tools were designed to support the creation of subspaces in an intuitive way. However, the definition and extraction of subspaces is only part of the process of investigating the complexities of an *n*-dimensional space via visualization.

In this chapter we discuss the second part of the investigation procedure which consists in organizing the created subspaces in such a way as to support navigation in the information space. We explain our proposed navigation mechanism aimed to reduce the chances of the user getting lost during the exploration of the *n*-space, and to encourage the user to explore the high-dimensional space until the visualization goal is achieved.

5.1 Exploration Issues in High-dimensional Space

Strothotte *et al.* [188, Chapter 2] observed that "navigation in complex information spaces requires a careful structuring of the information space." To provide a uniform structure for the treatment of high-dimensional space that supports navigation we first need to acknowledge the differences that exist between multivariate and multidimensional data. So far we have compared the inherent nature of both multivariate and multidimensional data

spaces (c.f. Chapter 1, Section 1.1.6) and how their distinct organization influences the way the filter process is applied to the *n*-space (c.f. Chapter 3, Section 3.1.1). This very distinction also affects the way the exploration of the *n*-space is conducted.

At the centre of the uniform visualization treatment we seek is the concept of a *cell*. A cell is the building block of the organizing structure that we impose on the high-dimensional data space to create a visualization. There are two types of cell, depending whether the cell is extracted from a multivariate or a multidimensional dataset. A cell extracted from a multidimensional *data space* is a 'subspace' in the real sense of the word. Truly a cell extracted from a multidimensional dataset is a 'slice' or 'piece' of the data space through a specific location – the focus point – within the *data space*. On the other hand, a cell extracted from a multivariate *data space* could be regarded as a 'subset' rather than a proper subspace. This 'subset' is the result of a projection of the multivariate data elements onto a low-dimensional geometric space (i.e. a line, plane or 3D region) in the *visual space*. The projection to form a multivariate cell is accomplished by dropping off the variates that are not part of a that cell.

These different interpretations of a cell, either as a 'subspace' or as a 'subset', is the first factor in determining a navigation procedure. In both cases the focus point determines the location where the cell is extracted from. For the multidimensional case the positioning of the focus point is crucial, because different locations yield different slices; while for the multivariate case the location of the focus point determines the location of the projection 'plane', but because an orthogonal projection is employed (i.e. simply ignoring the variates that have not been selected to form a cell) the final result is not affected by the location of the projection 'plane'[1].

The existence of these two different views of a cell is caused by the discrete nature of the data models usually associated with multivariate datasets. Early on in Chapter 1 (Section 1.1.6) we mentioned that the *empty space phenomenon* predicts that multivariate spaces of increasing dimensionality generated by finite samples tend to be empty, therefore there is no point in trying to visualize the whole space. If we did so we would usually see an almost empty space with a few data items scattered around. It seems to be more productive to concentrate on visualizing only the regions of that space where the data points are located – hence the common treatment of a multivariate dataset as a 'table' of data items whose columns represent variates and whose rows represent data elements, rather than treating the variate space as a multidimensional data space.

[1] One of the proposed topic for future work presented in Chapter 8 (Section 8.3) involves looking into different projections (e.g. perspective projection, or distorted projections) when extracting cells from multivariate data, aiming to investigate whether this would promote any gain of new insight.

Normally the investigator already has a prior knowledge of the relevant combinations of variables that make any sense to the problem in focus – this is the second factor in determining a navigation strategy. In the next sections we discuss this topic: the investigator's prior knowledge of the subject under investigation. We then organize it into possible investigation scenarios, describing how these scenarios may be tackled using *HyperCell*.

5.2 Exploration Challenges

The main exploration objective is to support the navigation of *n*-space to achieve the visualization goals. The central element in the exploration procedures is the cell. The total number *m* of possible one-, two-, and three-dimensional cells added together ($m = \frac{n(n^2+5)}{6}$) is too high to allow the straightforward strategy of generating them all. Creating all possible cells for a dataset with a large value *n* would require considerable rendering power, possibly in a parallel configuration, for the user to have a real-time response from the visualization system. Furthermore the screen space is at a premium and cannot be wasted with information that might not be relevant for the visualization process.

Nonetheless we need to, somehow, make it clear for the user that a great number of cells exist and may be instantiated if required. Additionally, a data visualization technique should try to offer an overview of the data, ideally at the start of the exploratory process (according to the principle of Visual Information Seeking Mantra by Shneiderman [176] – *overview first, zoom and filter, then details-on-demand*).

Therefore the first challenge is to suggest a lay-out organization for the cells that would allow the user: (a) to establish some relationship (possibly continuity) among neighbour cells; and (b) to have an idea of the whole array of cells that can be created. This lay-out scheme should be associated to the entity that functions as a 'container' for the created cells – i.e. the *Workspace Manager*.

The exploration also involves the maintenance of some sort of recording mechanism to keep track of any changes made to the *n*-dimensional window of interest. Changing the parameters of the *n*-dimensional window is the main exploratory activity. In the particular case of multidimensional data we would also need to keep track of the translations of the *n*-dimensional window's focus point. Therefore the final challenge is to devise a logging device that affords a visual representation for the changes made to the the *n*-dimensional window.

5.3 Investigation Scenarios and Exploration Strategies

One basic assumption is that the objective of the exploration has to be clearly determined beforehand. Here is just a short list containing examples of tasks and visualization goals related to high-dimensional applications, gathered from the literature within the visualization field:-

- *Understand scalar function of several variables.* This may involve finding locations in *n*-space (function domain) where the function assumes a specific value or range of values (e.g. isosurfacing in the subspaces); investigating a sub-region around local or global *extrema*; or, verifying correlations between function values and domain values [10, 199].

- *Search for patterns and/or structures*. A common goal in the investigation of multivariate datasets is to look for hidden patterns within the data [37], for example to recognize special and important structures such as clusters [209] and outliers [12]. Outliers, in particular, are important because they could be responsible for a misinterpretation of the entire dataset if not adequately treated.

- *Comparison of subsets of data and discovery of relationships*. Sometimes a data visualization method may be used solely to aid the comparison of data items according to a particular variate or set of variates [158]. Projections of high-dimensional data onto low-dimensional subspaces usually provide insightful views, aiding the understanding of multivariate relationships [76]. The most commonly sought relationship is correlations between variates, which is usually revealed through a mechanism such as *brushing and linking* [98].

- *Visualize the result of a complex simulation with a large number of parameters*. Often intricate phenomena are simulated in a computer via complex mathematical models involving several parameters. Understanding the trade-off between different combinations of parameters and the corresponding outcome is possibly an important objective [155]. This type of knowledge surely helps the theory formation process, which in turn might describe or predict the phenomenon that generated the dataset [197].

- *Data exploration*. Frequently after a complex dataset has been acquired, investigators want to browse and explore the data without a clear goal, just to obtain a preliminary impression of its content [58]. The data, for instance, may have an

inherent hierarchical organization and the intent would be to better understand the structure [74, 218].

Once a clear aim has been established it is possible to identify four scenarios describing the explorer's knowledge about the object of study, as depicted in Table 5.1.

WHERE	**WHAT**	
	Knows	*Does not know*
Knows	(1): User knows what s/he is looking for and where to find it.	(2): User does not know what s/he is looking for but has an idea of interesting places to look at.
Does not know	(3): User knows what s/he is looking for but has no idea of its whereabouts.	(4): User has no clue on what s/he is after neither where to start looking for.

Table 5.1: Listing four possible exploratory scenarios based on the prior degree of knowledge about the object of investigation.

Each scenario described in Table 5.1 requires a different approach. Below we offer four different strategies designed to address each of those scenarios. The objective of suggesting these strategies is to guide the exploration of the *n*-space towards the visualization goal.

- **Scenario 1:** Possibly the best approach for this case is to move the *n-dimensional Window* directly to the location of interest. The *n-dimensional Window*'s ranges may be set to cover the region under investigation that contains the interesting features. Setting the *n*-dimensional window to cover only the region of interest has the benefit of avoiding either the calculation or the loading of data that is not of interest.

  If an evaluation module is available and associated with the data it is reasonable to request a high sampling rate, thereby making more detail available in the visualization.

  Once the user has adequately set up the *n-dimensional Window* over the desired location, it is possible to examine the area surrounding that location by requesting the generation of the specific projections that are expected to convey meaningful information.

- **Scenario 2:** This scenario resembles the previous one in the sense that the explorer knows where to start the exploration but is not sure about what to expect there.

  A reasonable approach is to start as in *Scenario 1*, setting a precise focus over the expected locality of interest, and then extract all possible 3D cells to obtain

an overview of the area. If the number of variables is high this will cause the generation of an excessive number of 3D cells, hence an alternative option is to request the creation of all possible 2D cells instead. In this case fewer cells will be generated and, consequently, less space on screen will be required. Ultimately if the number of dimensions is too high even for 2D cells, we recommend the use of the Dynamic Cell mechanism to probe that location until an appropriate number of meaningful projections are identified.

In the particular case of multivariate data a valuable strategy is to select the projections (i.e. set of variates) that are expected to convey meaningful information, and consecutively map the remaining variates to the *graphical properties* of the Visual Mark, in a type of 'attribute animation'. Such strategy helps to bring out possible correlations between the variates of the cell and the other variates not included in the cell.

- **Scenario 3:** It is possible to address this case with two different strategies. In the first one a pre-processing step (in the *data analysis* stage) could be applied to identify, for instance, the location of *extrema*; or simply use the *data analysis* stage to locate regions in which a specific value is found. Once a target location is defined one should position the *n*-dimensional window over the target location and explore the area following the suggestions for *Scenario 2*.

  A second strategy is to to generate sets of 'disjoint' cells. By disjoint we mean cells that have no variable in common. So, for instance, if we have an eight-dimensional data space we would have two disjoint 4D cells: *cell-(1,2,3,T:4)* and *cell-(5,6,7,T:8)*. Once the disjoint cells are instantiated the exploration could be done simply by animating the focus point along the coordinate of the dimension set as time; or animating the *graphical properties* of the Visual Marks through the remaining variates, in the multivariate case. If no interesting feature is found in any of the cells, the user should try a different combination of variables to form the disjoint cells. If something relevant comes to attention then a potential interesting place has been found and we are back to the situation covered in *Scenario 2*.

- **Scenario 4:** In this case it is reasonable to try to identify some data analysis procedure that might bring out clues towards an interesting location. A controlled search can be made as in the second strategy of *Scenario 3*, using four-dimensional disjoint cells to cover a larger area in a short time.

Alternatively, one may simply use the Dynamic Cell to probe one location, mark that spot as visited and then move to another location. This may be more efficient because the user deals with only one visualization at a time. Consequently the investigation is faster than looking at several static cells because the user attention is focused only on one cell and there are fewer parameters to control.

In summary there are three methods of exploring the *n*-space:

1. To use a Dynamic Cell to probe the *n*-space, which is done within the same *n*-dimensional window (c.f. Chapter 4, Section 4.4.1). Probing the *n*-space is accomplished by selecting different variables while observing the visualization changing.

2. To generate several cells and dynamically change the *n*-dimensional window configuration through the *n-dimensional Window* tool; then observe the effect that this action has upon all cells linked to that *n*-dimensional window. This operation can be compared to 'moving' several views of the *n*-space simultaneously, as we move the focus point.

3. To create several filters to investigate different points at the same time. This is a valuable feature if the visualization goal is comparison. This strategy is especially useful for multidimensional applications, where one wishes to compare different locations in *n*-space. It could also be applied to multivariate data, for instance, in a situation in which filters are applied in parallel and the results combined to emulate an OR operation between variates (this is similar to the disjunctive queries discussed in Section 2.4.3.1, Chapter 2).

Next we describe the tools that support the exploration of the *n*-space, organized by the three different methods of exploration described above.

5.3.1 Method 1: Probing the *n*-Space

Probing the *n*-space is done through the use of the Dynamic Cell mechanism, introduced early on in Chapter 4, Section 4.4.1. To recap, by clicking and selecting/unselecting vertices from the *Interaction Graph* the user is able to observe in the same view the progressive cell transformation from a simple low-dimensional configuration up to a 3D cell with animation, and vice versa.

The Dynamic Cell is a potentially useful feature of *HyperCell* because when the tool is used to increase progressively the dimensionality of a cell it reveals a certain degree of continuity. This happens because when we add a variable to an existing cell we are

'expanding' that cell in a new dimension and the final result is a cell whose subspace contains the initial cell.

5.3.2 Method 2: Manipulating the *n*-dimensional Window

Since the introduction of the filtering strategy as the basis for *HyperCell* we have interpreted a filter as a window located in *n*-space, and the cells are representations of subspaces that are 'views' through that window. In this context we could think of the *n*-dimensional window as an element defined in the *data space*, whereas the 'views' or subspace are elements defined in the *visual space*, derived from the *n*-dimensional window.

The correspondence between a filter device and a window in *n*-space has led us to realize that some operations usually associated with the concept of a window could be adapted in the filtering context to help understand the exploratory tasks. These operations can be applied either in the *data space*, thus affecting all 'views' associated with it; or in the *visual space*, thus affecting individual 'views' (i.e. cells). Below is a list of such operations:-

- *Window transformations:* This group includes operations such as *translation*, *scaling*, and *rotation*, which are implemented in the data space.

  Translation in this context means displacing the whole *n*-dimensional window W – i.e. the lower limits $L = (l_1, l_2, \ldots, l_n)$, the upper limits $U = (u_1, u_2, \ldots, u_n)$, and the focus point coordinates $F = (f_1, f_2, \ldots, f_n)$ of each variable i – by a fixed distance in a given direction $\vec{d}$ in *n*-space: $W' = W + \vec{d}$, where W is defined by (L, F, U). Of course, this operation only makes sense if the filter does not cover the entire data space domain.

  Translation would be accomplished simply by interacting with the *n-dimensional Window* diagram, and providing a second (destination) focus point F', thereby defining a direction $\vec{d} = (F' \vec{-} F)$.

  Scaling by an array of factor $S = (s_1, s_2, \ldots, s_n)$ means expanding ($s_i > 1$) or contracting ($0 \leq s_i < 1$) the *n*-dimensional window W towards the directions defined by the individual spokes that represent the variables i; and the scaling is centred on the focus point $F = (f_1, f_2, \ldots, f_n)$, which remains unchanged after the transformation.

  Therefore the scaling of the individual components of the lower and upper limits can be expressed, respectively, by the pair of expressions: $l'_i = f_i + (l_i - f_i)s_i$ and $u'_i = f_i + (u_i - f_i)s_i$.

Note in particular that setting a window smaller than the data space is a valuable tactic if the data space is too large to be visualized in its totality or too demanding to be generated throughout the domain.

Rotation in this context means rotating the window in *n*-space around the focus point, which is different from rotating the views (i.e cells) in the *visual space*.

Note that in the *n*-space there are $C_2^n = \frac{1}{2}n(n-1)$ distinct pairs of coordinate axes (degrees of freedom) that can be used as the plane of rotation[2]. One way of generating a matrix that implements a rotation around a given plane in *n*-space is to multiply a sequence of matrices each one representing a rotation in the plane defined by a given each possible pair of canonical axes.

For example, any 4D rotation around the origin can be implemented by the following matrix concatenation

$$M = R_{1,2}(\theta_{1,2}) \cdot R_{1,3}(\theta_{1,3}) \cdot R_{1,4}(\theta_{1,4}) \cdot R_{2,3}(\theta_{2,3}) \cdot R_{2,4}(\theta_{2,4}) \cdot R_{3,4}(\theta_{3,4})$$

where a matrix $R_{i,j}(\theta_{i,j})$ implements a rotation by an angle $\theta_{i,j}$ in the plane defined by the pair of axes $(x_i, x_j), i, j = 1, \ldots, n$. A matrix $R_{i,j}(\theta_{i,j})$ has the following rule of formation: (1) all the elements $(e_{l,k}, l = k)$ from the main diagonal are equal to 1, except the $e_{i,i}$ and $e_{j,j}$, which are equal to $cos(\theta_{i,j})$; and (2) all the remaining elements $(e_{l,k}, l \neq k)$ are equal to zero, except $e_{i,j} = -e_{j,i} = -sin(\theta_{i,j})$.

Probably the exploration process could benefit from the rotation operation if we consider the Principal Components as the destination axes for a rotation sequence. Although we acknowledge that rotation in *n*-space may be a valuable tool (especially when searching for interesting projections of multivariate data), it has not been implemented in *HyperCell* because of the time constraints.

- *Changing resolution:* This is an operation performed on the *visual space* to control the view's resolution (i.e. the number of samples in each variable of a cell). Different resolution controls the level of detail in the visualization of a cell. This may be useful, for example, in a situation in which the user requests the visualization of a large amount of cells, just to get an overview of the whole space. In this case it is sensible to render the cells initially at a low resolution, identify the cells of interest and then request them to be rendered at a higher resolution to obtain more detail.

[2] For more detail on this see for example the introduction to rotation in *n*-space given by Hanson in [90].

This feature is implemented in the *Subsetter* module, which provides a slider to control the sampling ratio.

5.3.3 Method 3: Multiple Filters

The paradigm that has been described so far is one of sequential exploration of high-dimensional datasets, through successive low-dimensional subspaces, with a smooth transition between these subspaces. This can be extended to allow multiple filters or concurrent views, where it is possible to retain views of where we have visited in our previous explorations.

This extension is encouraged by the fact that the *HyperCell* method has been implemented in a modular fashion. This naturally permits an expansion of functionality by simply creating copies of the pipeline that implements the filtering process. This is exactly the recommended exploration strategy for the case in which the high-dimensional dataset is very extensive or a comparison of distinct *loci* in *n*-space is required.

The filter core functionality is implemented by three modules, IGraph, NDWin, and Subsetter, as shown previously in the scheme of Figure 4.1 (Chapter 4). This model can be naturally expanded in the way shown in Figure 5.1, where visualization now accommodates multiple Filtering steps, each one associated with a *Workspace Manager* that maintains a record of all cells that have been created. Therefore the output from the Filtering steps is now an *array* of Focus Data.

A nice feature of this approach is that dynamic changes to a given *n*-dimensional window are propagated to all elements of the *Workspace Manager* associated with it, and so all visualizations (i.e. Focus Data) linked to it are dynamically altered. The experience may be compared to walking through the *n*-dimensional space (by moving the focus point for example) and seeing the effect in all the subspaces previously created – in other words, to look around in different directions in the *n*-dimensional world. These dynamic changes are a nice application of the 'Snap' visualization concept introduced by North and Shneiderman [150], which recommends that systems employing multiple views should enable users to rapidly and dynamically compose coordinated visualizations in an attempt to support exploration, preferably without programming.

Multiple filter processes allow the behaviour in multiple *n*-dimensional windows to be studied simultaneously, and have the advantage of aiding us to keep track of all visited locations in *n*-dimensional space. This capability addresses one of the disadvantages of the *hyperslice* technique, listed earlier in Section 2.4.2.4: *hyperslice* only supports the creation of a single focal point, thereby hindering any chance of comparing distinct *loci*

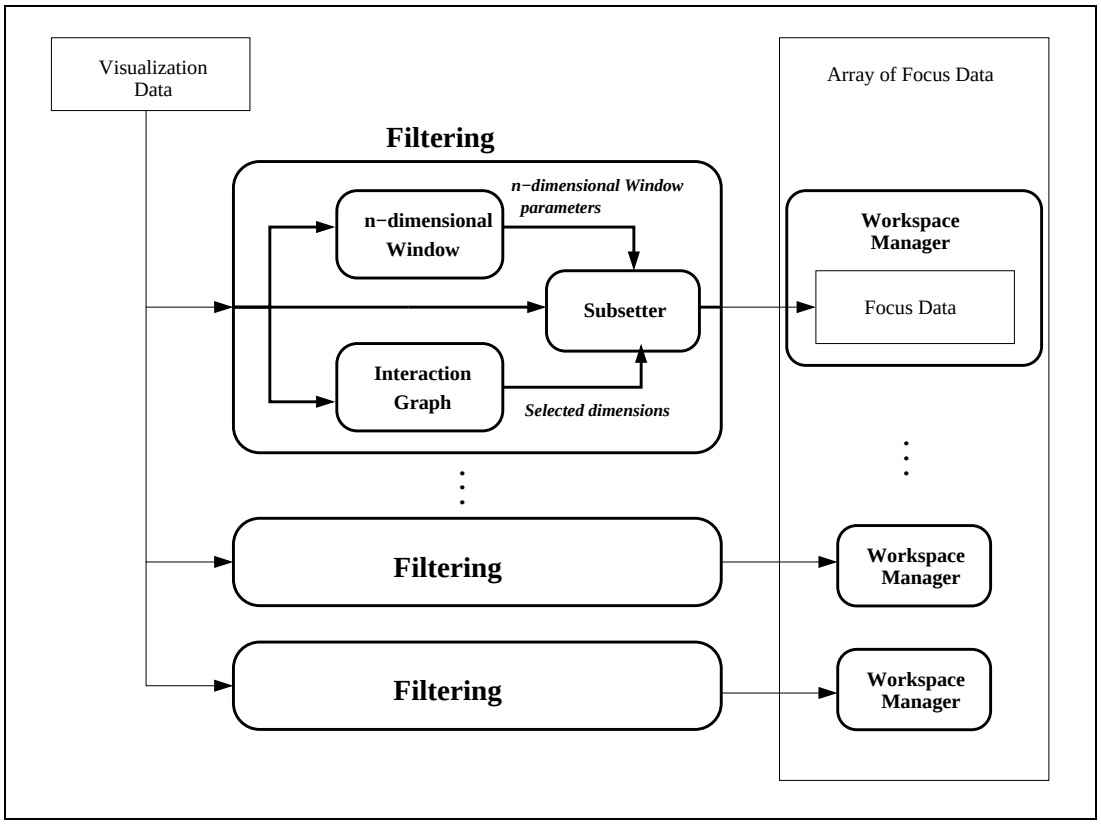


Figure 5.1: An extended version of the filtering model to allow multiple filters or concurrent views corresponding to the visited locations in the *n*-dimensional space. The final output is now an 'array' of Focus Data.

in *n*-space. In addition, multiple filters in parallel can be applied to a multivariate dataset to emulate the OR logical operation between variates.

All the operations mentioned so far are realized through the tools that have already been introduced, namely *Interaction Graph*, *Subsetter*, and *n-dimensional Window*. In the next two sections we discuss the coordination and organization of multiple cells and the navigation mechanism designed to support the exploration of *n*-space.

5.4 *Workspace Manager* – Coordinating the Subspaces

Early on in Section 5.2 we mentioned that one of the major challenges for the exploratory task is to provide the user with some device that represents the set of all possible cells. The disposition of the cells should follow an arrangement algorithm that transmits a feeling of organization and possibly connection between cells. In addition, we have already mentioned that the filtering process can be greatly enhanced if coupled with a multiple-

coordinated-views scheme whose views are somehow ‘linked’. Providing a connection between multiple views enables users to have the sensation that those views are part of a whole, thus representing the same high-dimensional entity. Our suggested solution for this challenge is a tool called *Workspace Manager*, whose interface is shown in Figure 5.2.

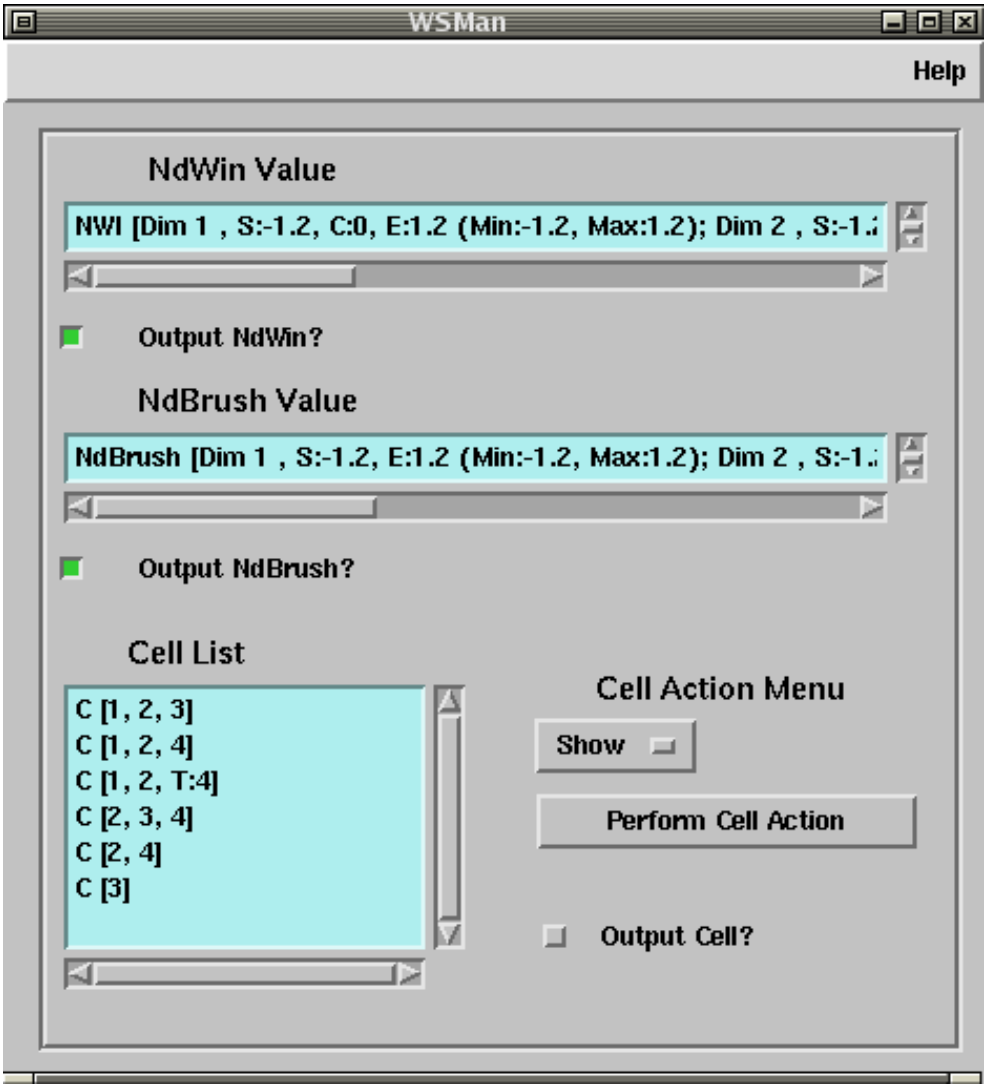


Figure 5.2: *Workspace Manager* user interface. There are textual representations for both *n-dimensional Window* and *n-dimensional Brush*, located at the top region of the interface. Below that there is a list box widget containing the list of all cells that have been created so far, ‘through’ the filter associated with this workspace.

The primary function of the *Workspace Manager* is to centralize all the information that describes a filter (i.e. the parameters from the *n-dimensional Window*) and the list of subspaces that have been extracted using the filter. Therefore this module has to be connected to the NDWin (the module that implements the *n-dimensional Window*), the

IGraph (the module that implements the *Interaction Graph*), and also the NDBrush (the module that implements the *n-dimensional Brush*).

The *Workspace Manager* tool stores (a) the current configuration of the *n*-dimensional window; (b) the size of the *n*-dimensional brush; and, (c) the list of cells that have been created, sent by the IGraph. The first two pieces of information are transmitted downstream to all cells registered with the *Workspace Manager* whenever they change as a result of user interaction.

The secondary function of the *Workspace Manager* is to provide graphical representations of the lay-out organization imposed on the created cells. This lay-out should be capable of accommodating all possible cells. Our attempts towards such arrangement of cells are discussed next.

5.4.1 Lay-out of Cells

A continuing theoretical challenge is the combinatorial problem of selecting groups of 3 from *n*, in such a way that all possible selections are covered and organized on screen in an optimally 'spread out' fashion. Why is this important? It is a good strategy to provide the user with an overview of the entire high-dimensional space, and one way to accomplish this is to generate all possible combinations of subspaces automatically.

The automatic generation of cells is done by the IGraph module, which was described earlier in Section 4.4.3 (Chapter 4). In that section we draw attention to the fact that the generation of all possible combination of variables creates a great number of cells whose display could rapidly overwhelm a user. The created cells ought to have an underlying organization, ideally one that communicates some sense of connection between subspaces.

In the next two sections we discuss two different arrangements of cells, one aimed at presenting two-dimensional cells in a way that minimizes the use of space on screen while showing all variables; and another designed to accommodate all possible cells into a single structure.

5.4.1.1 First arrangement: the 'Fruit machine' layout

The initial objective was to find an arrangement in which all possible combinations of variables, without repetition of elements, would be generated. One good strategy to generate all possible combinations of two integer numbers was proposed by Brodlie in [24], as part of a minimization algorithm. He compared this task with the fixture strategy problem in which one needs to "find a fixture list for *n* teams meeting in pairs, each team meeting each other exactly once in $(n-1)$ weekly stages."

The solution procedure is described as follows:-

1. Organize the numbers in pairs, forming a *block*. Below we have an example of an initial block, for the case $n = 6$. (Note that the number are ordered clockwise, throughout the block.)

$$(1) \begin{bmatrix} \mathbf{1} \\ 6 \end{bmatrix} \begin{bmatrix} 2 \\ 5 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

2. Each new block is generated by permuting every number clockwise, except number **1** that shall remain in its position during the whole procedure. The application of this procedure to the above starting block configuration is shown next.

$$(2) \begin{bmatrix} \mathbf{1} \\ 5 \end{bmatrix} \begin{bmatrix} 6 \\ 4 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad (3) \begin{bmatrix} \mathbf{1} \\ 4 \end{bmatrix} \begin{bmatrix} 5 \\ 3 \end{bmatrix} \begin{bmatrix} 6 \\ 2 \end{bmatrix},$$

$$(4) \begin{bmatrix} \mathbf{1} \\ 3 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \end{bmatrix}, \quad (5) \begin{bmatrix} \mathbf{1} \\ 2 \end{bmatrix} \begin{bmatrix} 3 \\ 6 \end{bmatrix} \begin{bmatrix} 4 \\ 4 \end{bmatrix}.$$

The nice feature about this procedure is that it always creates an arrangement that has the same number of pairs in each block, thus occupying an even space on screen. Also this arrangement distributes the numbers so that each pair of numbers appears exactly once in each block. Contrast this with a matrix-like arrangement used, for instance, in a two-dimensional scatterplot matrix. For this particular case, $n = 6$, we would end up with a 'staircase' arrangement having at the base 5 pairs, then 4, 3, 2, and 1 pair at the top (of course we are ignoring the main diagonal and the upper triangle of the full matrix). Thus the use of screen space is not as optimal as for Brodlie's approach[3].

This strategy for pairing integers provides the basis for the first proposed arrangement involving two-dimensional cells, where the integer numbers are associated to variables. Initially, all blocks (in the previous example there are 5 blocks) are generated and displayed simultaneously in a vertical sequence. The result is an even occupation of the display area, showing all possible combinations of two-dimensional cells. If the screen space is limited and showing all possible 2D cells is not an option, we can show one block at a time, simulating the behaviour of *fruit machine*[4] (except that the selection of numbers is not random), in which pressing a button on the interface would display the next block of 2D cells.

[3]Note that if n is odd the procedure is almost similar, the exception being that the number **1** shall move clockwise in the same way as the other numbers.

[4]A coin-operated gambling machine that produces random combinations of symbols (usually pictures of different fruits) on rotating dials; certain combinations win money for the player.

THE SCHOOLGIRLS' WALK

The next natural step was to look for a similar algorithm to handle the distribution of n in triplets, having the same important feature of exploiting the screen space in an optimal way. For this particular instance the total number of possible triplets is given by $C_3^n = \frac{1}{6}n(n^2 - 3n + 2)$. One approach we have pursued was based on the solution for *the schoolgirls' walk* combinatorial problem proposed by Reverend Kirkman [112] in 1850.

The challenge is described as:-

> *A school mistress has fifteen girl pupils and she wishes to take them out on a daily walk. The girls are to walk in five rows of three girls each. It is required that no two girls should walk in the same row more than once per week. Can this be done?*

Solving this problem would provide us with a solution for a particular case of our problem ($n = 15$). But we expected that this solution could be generalized to solve a similar problem for any value n. However, at the end of our search for a solution we realized that combining triplets in *blocks* without repetition of pairs is a complex combinatorial problem whose solution in itself requires the use of complex algorithms, as described in [53]. Consequently we decided to abandon this strategy, and look for an alternative way of organizing the cells.

5.4.1.2 Second arrangement: The 'Building' lay-out

A simple way of systematically combining triplets of variables, represented by integer numbers, is to use the algorithm presented in Figure 5.3.

```
int i, // First variable coordinate.
    j, // Second variable coordinate.
    k, // Third variable coordinate.
    n; // Total number of variables.

for ( i = 1; n <= n-2; i++ )
    for ( j = i+1; j <= n-1; j++ )
        for ( k = j+1; k <= n; k++ )
            cout << i, j, k << endl; // Prints triplet.
```

Figure 5.3: Triple Nested loops algorithm for generation of triplets.

The outcome of the *triple nested loop* algorithm for an increasing number n of variables has an interesting pattern. In Table 5.2 we have listed the output for the triple nested

loop algorithm, separating in columns the executions for different values of *n*, and grouping in rows the triplets with same first variable.

Floors	$n=3$	$n=4$	$n=5$	$n=6$
1	123	123 **124** **134**	123 124 **125** 134 **135** **145**	123 124 125 **126** 134 135 **136** 145 **146** **156**
2		**234**	234 **235** **245**	234 235 **236** 245 **246** **256**
3			**345**	345 **346** **356**
4				**456**
Total	1	4	10	20

Table 5.2: *Triple nested loop* style of generating all possible combination of cells. The triplets in bold are the new additions as a result of running the algorithm for the next value n.

We use a metaphor of a building to describe the arrangement resulting from the triple nested loop algorithm. In this metaphor a triplet represents a *room* in a building. The room is uniquely located in this manner: the first element of the triplet indicates a *floor* (thus the label floor used in the first column in Table 5.2); the second element indicates a *corridor*; and, the last element indicates the *room's number*. Figure 5.4 shows the *building arrangement*.

Here are some mathematical relations in a *building arrangement*:-

1. Building size is indicated by an integer $n \geq 3$.
2. Range for a room r: $3 < r \leq n$.
3. Range for a corridor c: $1 < c < n$.
4. Range for a floor f: $1 \leq f < (n-1)$.
5. Number of corridors on a floor f of a building n is: $C(n, f) = n - (f+1)$.
6. Number of rooms on a floor f of a building n is: $R(n, f) = \frac{C(n,f) \cdot C(n,(f+1))}{2}$.
7. Number of rooms in a corridor c of a building n is: $R(n, c) = n - c$, which is independent of the floor f.
8. The top floor always has only one cell.

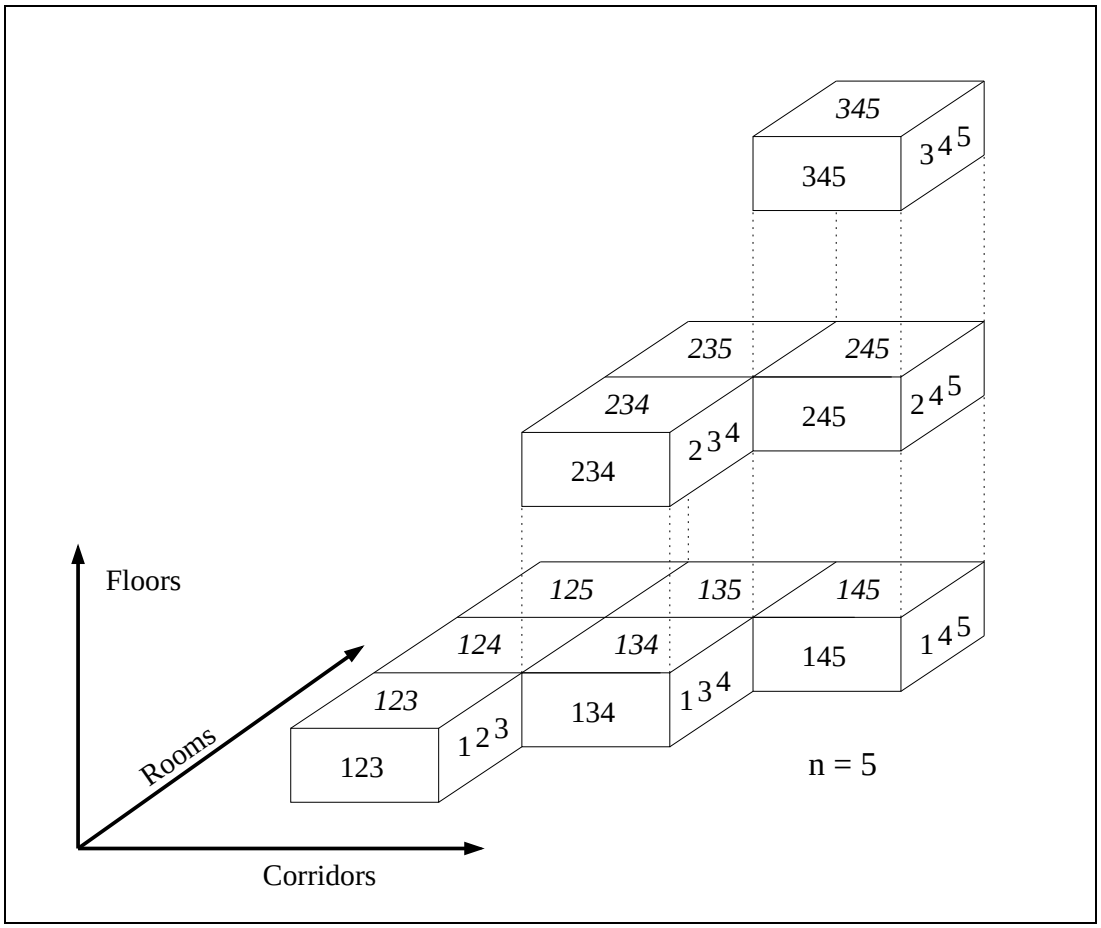


Figure 5.4: Metaphor of a building used to arrange cells. *Building* $n = 5$.

The nice feature of the *building arrangement* is that there is always a degree of continuity between neighbouring cells in the following sense: (a) movement within the same floor ensures at least one (the first) variable in common; (b) movement within the same corridor ensures exactly two variables in common; and, (c) corresponding rooms on 2 floors have 2 variables in common.

It is possible to represent the building arrangement in a two-dimensional form, called *building lay-out*. This is done simply by projecting the 3D building along the *rooms* axis. (Note you can do it along any axis.) The result is shown in Figure 5.5-(a). This idea can be extended to accommodate the 2D cells, which are interpreted as rooms on the ground floor (i.e no first variable). The result is shown in Figure 5.5-(b).

We can improve this lay-out by thinking of it as an $(n-1)$ x $(n-1)$ matrix of cells. In this case we could avoid wasting the space corresponding to the triangle of cells above the secondary diagonal of the matrix by using it to accommodate the 2D cells, as shown in Figure 5.6-(a). In the same picture we added an extra line at the bottom to accommo-

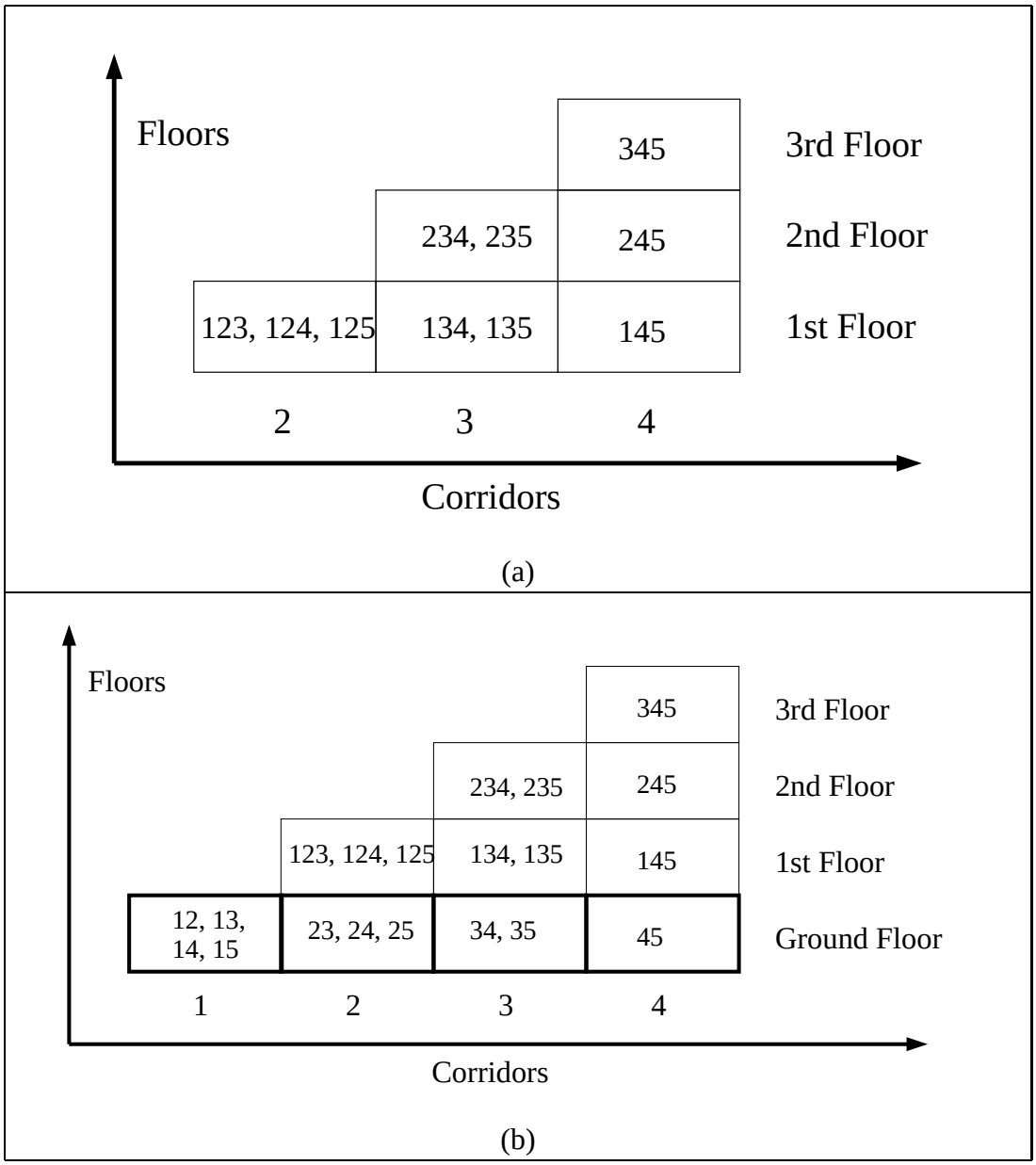


Figure 5.5: Projection of the building arrangement of Figure 5.4, along the *rooms* axis, creating a 2D representation.

date the 1D cells. The final version of the *building lay-out* could be the one shown in Figure 5.6-(b), in which the bottom line of cells used to accommodate the 1D cells are attached to the $(n-1)$ x $(n-1)$ matrix.

There are two advantages of using the *building lay-out*, the first one is that interaction mechanisms for 2D tends to be simpler than for 3D; and the second one is that using this two-dimensional user interface allows us to incorporate two- and one-dimensional cells as well. The *building lay-out* could represent all 1D and 2D cells as actual visualizations,

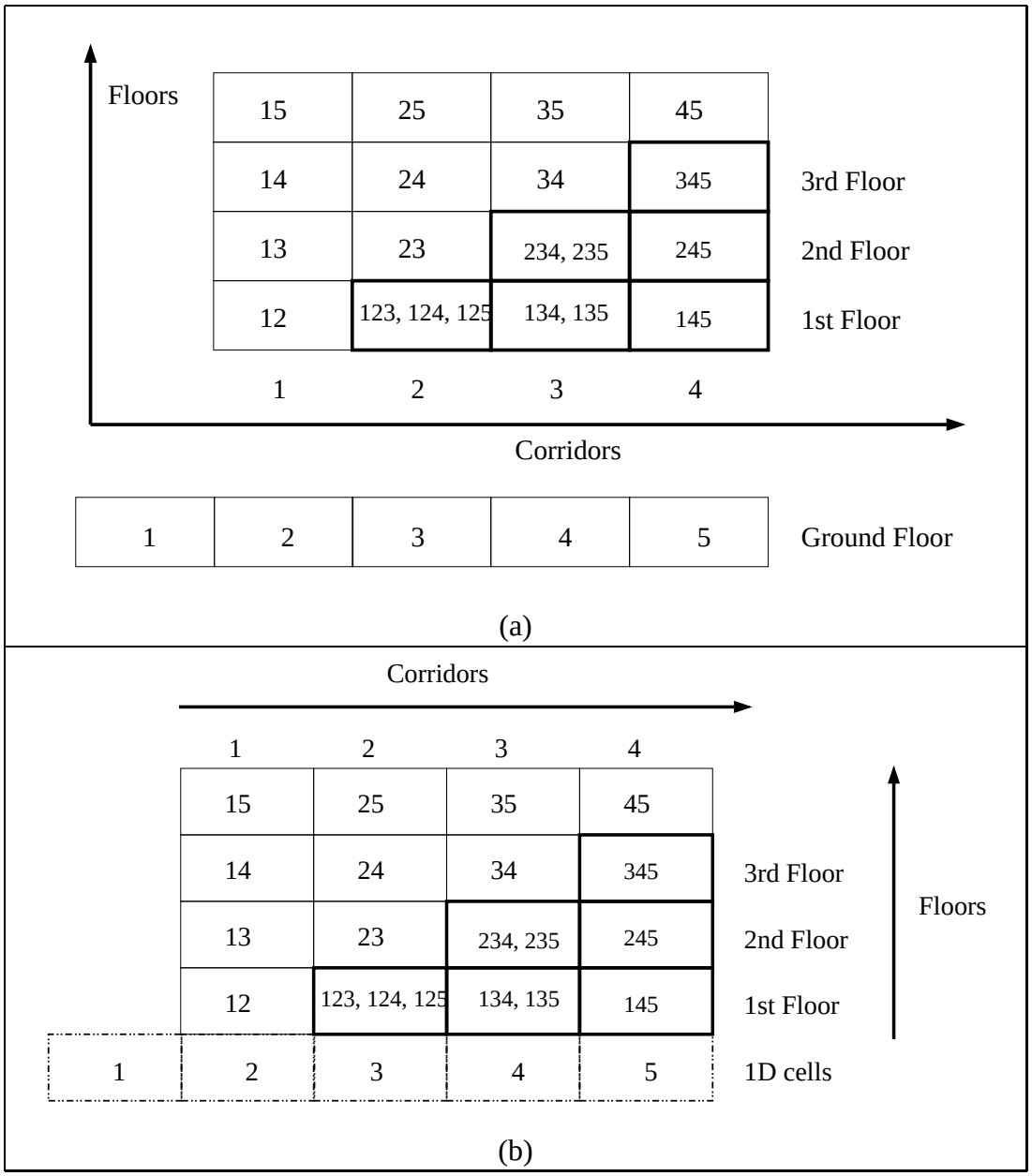


Figure 5.6: Expanding the building arrangement of Figure 5.5 to accommodate 1D/2D cells.

rather than just the cell numbers. In this case they could be done in low-resolution, i.e. using fewer samples to create the visualization, thus providing a fast rendering of the cells. The aim here is to offer the user an overview of the dataset. If a particular cell is of interest then the user clicks on it, triggering the generation of a full resolution visualization of the requested cell.

Regarding the 3D cells, they would not be represented as actual low-resolution visualizations because this would cause occlusion (note that sometimes there is more than one

3D cell in a ‘slot’ of the *building lay-out*). Rather they would be represented as cell numbers. Again, clicking on them would represent a request for a full resolution visualization of that 3D cell. Listing the 3D cells just as numbers tends to be a problem if the number of variables *n* is high, because of the limited space of a ‘slot’ in the matrix. To avoid this problem we can represent them in a smaller version of the *Interaction Graph* diagram, as shown in Figure 5.7.

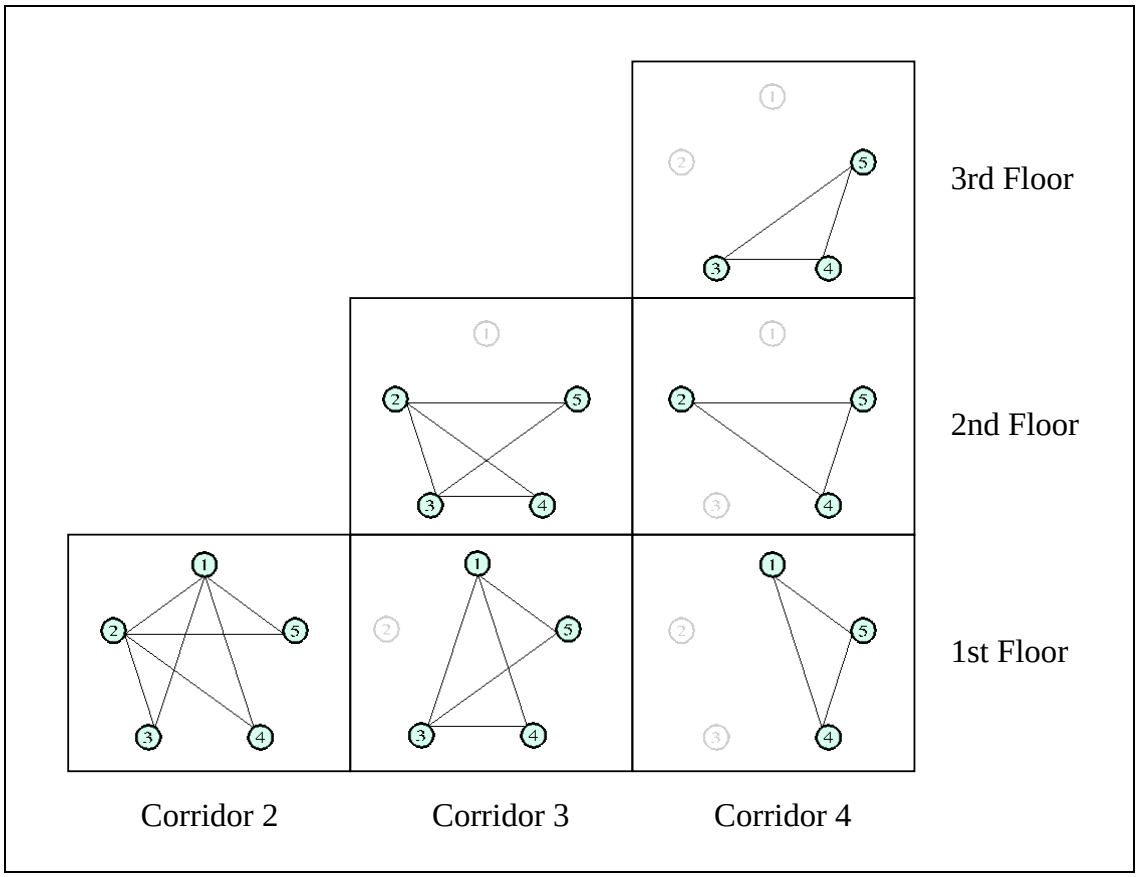


Figure 5.7: Part of the *building lay-out* that represents the 3D cells. The representation is done using the *Interaction Graph* metaphor.

A complete simulation of how the *building lay-out* should look is presented in Figure 5.8, on the interface of the *workspace manager* module. In that all 1D and 2D cells are represented by a low-resolution visualization.

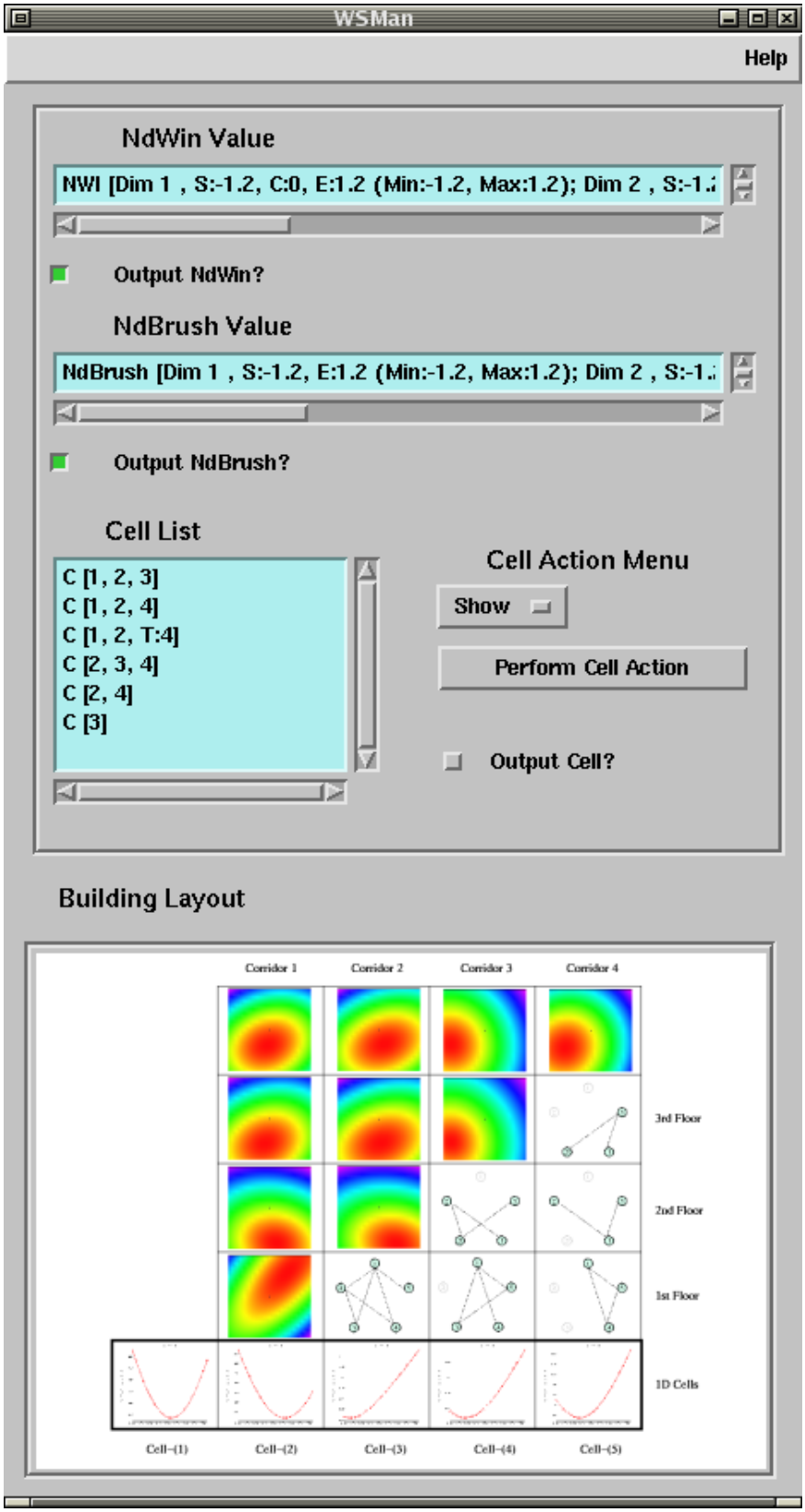


Figure 5.8: Interface of the WSMan module with the *building layout*.

5.5 *Navigation History* – Keeping Track of Navigation

The mental effort involved in investigating high-dimensional datasets and building a mental model of the data's underlying entity is considerable if the the dataset's dimensionality is large. Therefore it is very important to keep track of the interesting places that have been visited, and be able to re-visit such places if necessary.

To address this problem we have developed a module called NDNavigator to maintain a type of audit trail for changes made on the *n-dimensional Window*. The design of the *navigation history* visual tool draws upon the results of the GRASPARC project [26], where a tree representation was used to record the progress of an investigation conducted within the GRASPARC problem solving environment[5]. This idea of using a tree representation was later extended and made generic, becoming a module in the IRIS Explorer environment, called HyperScribe [215].

GRASPARC's *history tree* records both data and parameters. A branch – depicted as a line – represents the execution of the computational experiment, based on a set of parameters. The scientist then controls how frequently a 'snapshot' of the intermediate results should be taken (i.e. data + parameters) and perhaps visualized. Each of these intermediate stages of execution is represented as dots on a branch.

Occasionally a situation may arise that requires the scientist to stop the experiment (for example, as a consequence of inconsistent results, or even because some interesting feature has been identified). This is the situation where the *history tree* becomes useful. The *history tree* device allows the user to rollback to any recorded point in the running of the experiment (i.e. a dot on a branch). From there it is possible to re-start the execution using the recorded data (perhaps after fine-tuning certain simulation parameters), thereby saving precious execution time. The re-start of the simulation creates a tree branch, meaning a specific decision taken by the scientist.

For our problem, however, we would require a simpler version of the *history tree* concept in the sense that it is only necessary to record parameter values, not data. Basically we need to record two types of events related to the navigation procedure: (a) changes made to the the *n-dimensional Window*'s ranges – i.e. translation of the windows as described in Section 5.3.2; and, (b) movements of the *n-dimensional Window*'s focus point.

The *navigation history* tool starts with a small polygon representing the initial configuration of the *n-dimensional Window*. This *n*-sided polygon has an integer number at the centre indicating the sequence of modifications made to the *n-dimensional Window*, and the number of sides *n* is equal to the dimensionality of the dataset – thus making the

[5]This is a computational environment which integrates both visualization and the computation process that provides the data for the visualization.

polygon's shape similar to the *n-dimensional Window*'s. Every change made to any of the *n-dimensional Window*'s range and then passed downstream by the NDWin module may be recorded in the *navigation history* as a new polygon on the same branch. Changes made to the focus point location creates a new branch, parallel to the original branch. Therefore branches mean different locations in *n*-space, and polygons on the same branch mean adjustment of the *n*-dimensional window of interest on the same location in *n*-space. Figure 5.9 shows this concept.

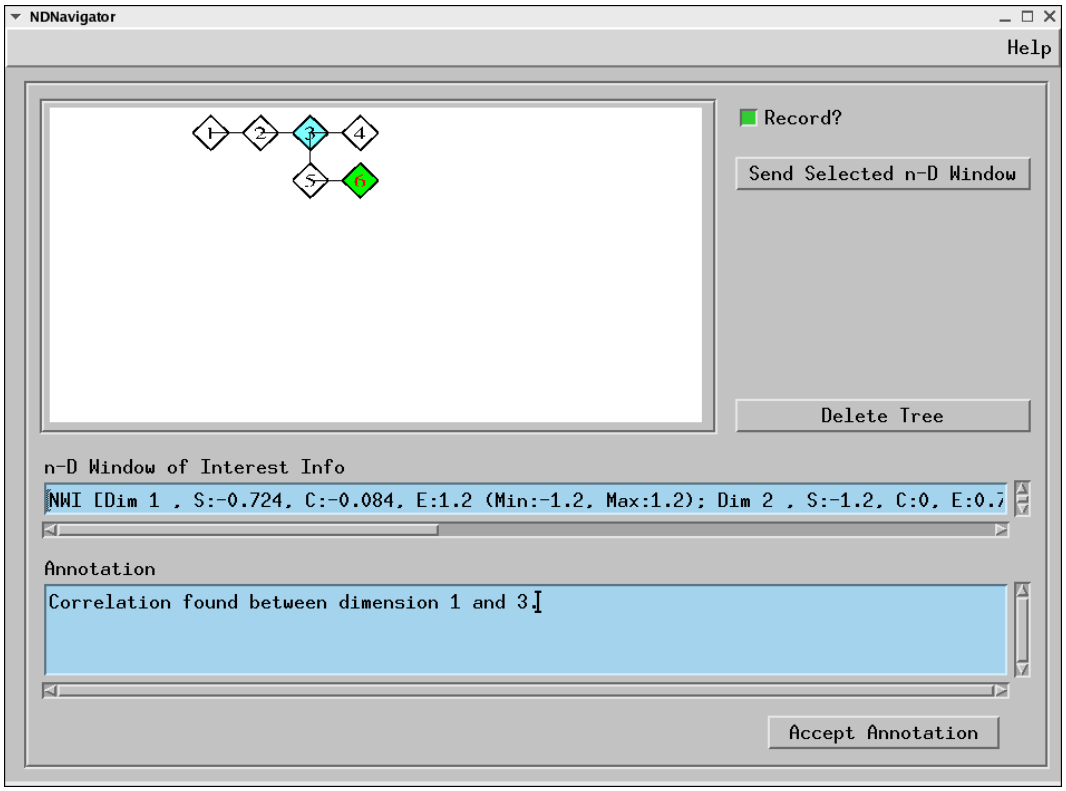


Figure 5.9: Graphical user interface of the NDNavigator, the module that implements the *navigation history* tool. Polygons 1 to 4 represent *n*-dimensional windows with the same focus point (same horizontal branch of the tree), whereas polygons 5 and 6 represent another focus point.

The NDNavigator (the IRIS Explorer module that implements the *navigation history* diagram) also allows the user to associate an annotation with any of the polygons of the tree, thus aiding the identification of interesting location for further investigation. Annotated polygons are represented with a cyan background (see polygon number 3 in Figure 5.9) and the polygon that represents the *current n*-dimensional window configuration is drawn with red foreground (see polygon number 6 in Figure 5.9).

Clicking on a polygon shows the corresponding annotation on the GUI of the module and makes that polygon the *selected* node; pressing the 'send' button (top right corner)

sends the stored *n-dimensional Window* information corresponding to the *selected* node back to the NDWin module, thus changing the current location in *n*-space accordingly and making the *selected* node become the *current* node.

An interesting improvement for the tree representation would be to map, for instance, the Euclidean distance between the current and the new focus point to the distance between branches. The only problem with this approach is that it might create overlapping branches, if a new focus point is too close to the original branch.

We have decided to implement our own navigation tree module instead of employing the HyperScribe module for two reasons. Firstly we wanted to keep a consistent interface between the *HyperCell* modules; secondly we wanted to incorporate specific functionality to the module, as follows: (1) the NDNavigator module uses a specific visual representation for the tree nodes, i.e. a polygon representing the overall shape of the original *n-dimensional Window* diagram; (2) the module encodes extra information on a tree node in the following manner – the order in which the data corresponding to the several *n*-dimensional windows has been sent to the module is represented by the numbers within each node; the presence of annotation associated with a tree node is depicted by the background colour of a node; and finally, the currently selected *n*-dimensional window is indicated by the foreground colour of a tree node; and, (3) the shape of the tree indicates if the navigation has been carried out at the same focus point (a single branch), or using different focus points (tree with several branches).

5.6 Summary

In this chapter we have covered the final stage in the process of visualizing high-dimensional data: the organization of cells and the navigation in *n*-space.

We have discussed the possible exploration scenarios depending on the degree of prior knowledge that the user may have about the target problem. For each of the possible scenarios we have suggested exploration strategies employing the several tools available to *HyperCell.* Basically we have described three exploration procedures: probing the *n*-space with Dynamic Cells, generating several views (i.e. cells) of a given location in *n*-space, or creating several filters to explore different locations in *n*-space.

Some of the challenges involved in providing a navigation mechanism for an *n*-dimensional data space have been discussed:-

1. The importance of presenting the cells in an integrated way, via a mechanism that organizes all possible cells in such a way that an overview of the data space is created and neighbouring cells are somehow related.

2. The need for a representation of the navigation performed.

We have tackled the first challenge by proposing and implementing a new module called *Workspace Manager*, whose function is to link together the cells created through a filter (i.e. *n-dimensional Window* plus *Interaction Graph*). The use of a workspace has two benefits: (a) changes made to the filter are propagated to all cells registered with the workspace; (b) the implementation of the workspace concept as a module encourages the creation of multiple filters that enables the visualization of different locations in *n*-space simultaneously.

Still related to the first challenge we have designed two solutions to address the problem of organizing the cells in a meaningful fashion. The first one is the 'fruit machine' layout, which organizes pairs of variables (i.e. a 2D cell) in a block in such a way that each variable appears exactly once in each block. The second solution is called *building lay-out*, which is based on a building metaphor. The *building lay-out* has the advantage of being able to accommodate all possible cells of different dimensionality and offers a degree of continuity between neighbouring cells.

For the second challenge we have proposed another tool called *navigation tree*, based on an early version of the GRASPARC's *history tree* concept. This mechanism keeps track of 1) changes made to the filter, i.e. changes to the filter's range and the translation of the filter throughout the *n*-space; and, 2) comments generated by users and associated with each node of the tree.

Chapter 6

Implementation Issues

THE PREVIOUS TWO chapters covered the issues related to the problem of visualizing high-dimensional data. We have tackled this problem by proposing a filtering process to reduce the complexity of the problem, followed by the coordinated visualization of the various 'filtered' data created in the process. In this chapter we describe how the elements that compose this filter mechanism and the subsequent visualization – introduced early on in Chapters 4 and 5 – come together to form the *HyperCell* visualization application.

This chapter is organized in a top down form. We first present an overview of the system, emphasizing the interaction between modules and how they relate to each other to form the basis for *HyperCell.* Then we move down to a more practical level, describing topics such as the implementation platform, programming languages used, the support classes created, and the idiosyncrasies of some modules.

The realization of such a system via implementation is an essential step because this allows us to assess the proposed designs and theories through the direct application to real world problems, which is the subject of the next chapter.

6.1 Software Specification

Here we list the basic requirements that guided the implementation phase of *HyperCell.* The following list has been defined as a result of the discussion that took place throughout

the previous chapters, and revolves around visualization, interaction, and insight issues.

1. *Compatibility with our proposed framework for high-dimensional data visualization.* The system should follow the design suggested by our framework in order to permit a uniform treatment of both multivariate and multidimensional data.

2. *Modularity and extensibility.* We are not proposing a novel geometric mapping for high-dimensional data, instead we are focusing on the filtering process that happens before the visualization mapping itself, and how this could be used to improve the level of understanding.

   The result of the filtering process applied to high-dimensional data is an 'array' of low-dimensional subspaces for which there already exist various well established visualization methods. Therefore we need a flexible system that enables the user to try out different visualization methods at the end of the filtering pipeline, where the several low-dimensional subspaces are generated.

   Another expected benefit from a modular system is the ability to create several filter devices by duplicating part of the system pipeline. This is important because it allows the system to be compatible with one of the exploration strategies described in Section 5.3 (Chapter 4), in which multiple filters are necessary to support comparison of different locations of the *n*-space.

3. *Multiple coordinate views.* The system should support multiple views of subspaces connected in a such a way that the user would be able to interpret them as parts of the same high-dimensional data space.

4. *Dynamic graphical user interface*. Most of the *HyperCell*'s functionality relies on a powerful graphical user interface (GUI) to afford the user control over the filtering process and the often required navigation within the *n*-space. Therefore the GUI needs to be capable of: (a) presenting a dynamic behaviour by reacting to the user interaction in such a way as to convey information/clues about the action that is about to be performed; and, (b) incorporating tight coupling aspects by indicating to the user the permissible operations according to the system's state, thus avoiding the execution of null actions.

5. *Efficient access to the data.* One of the advantages of using the filter approach is to avoid the need of having all the data available in memory for the visualization to happen. Therefore the ability to access or calculate only the pieces of information that have been 'filtered' is important to foster an approach that scales with the number of variables.

6.2 General Structure

In this section we present the overall structure of *HyperCell*'s components, depicting their interconnection and the exchange of data. Figure 6.1 sketches this structure.

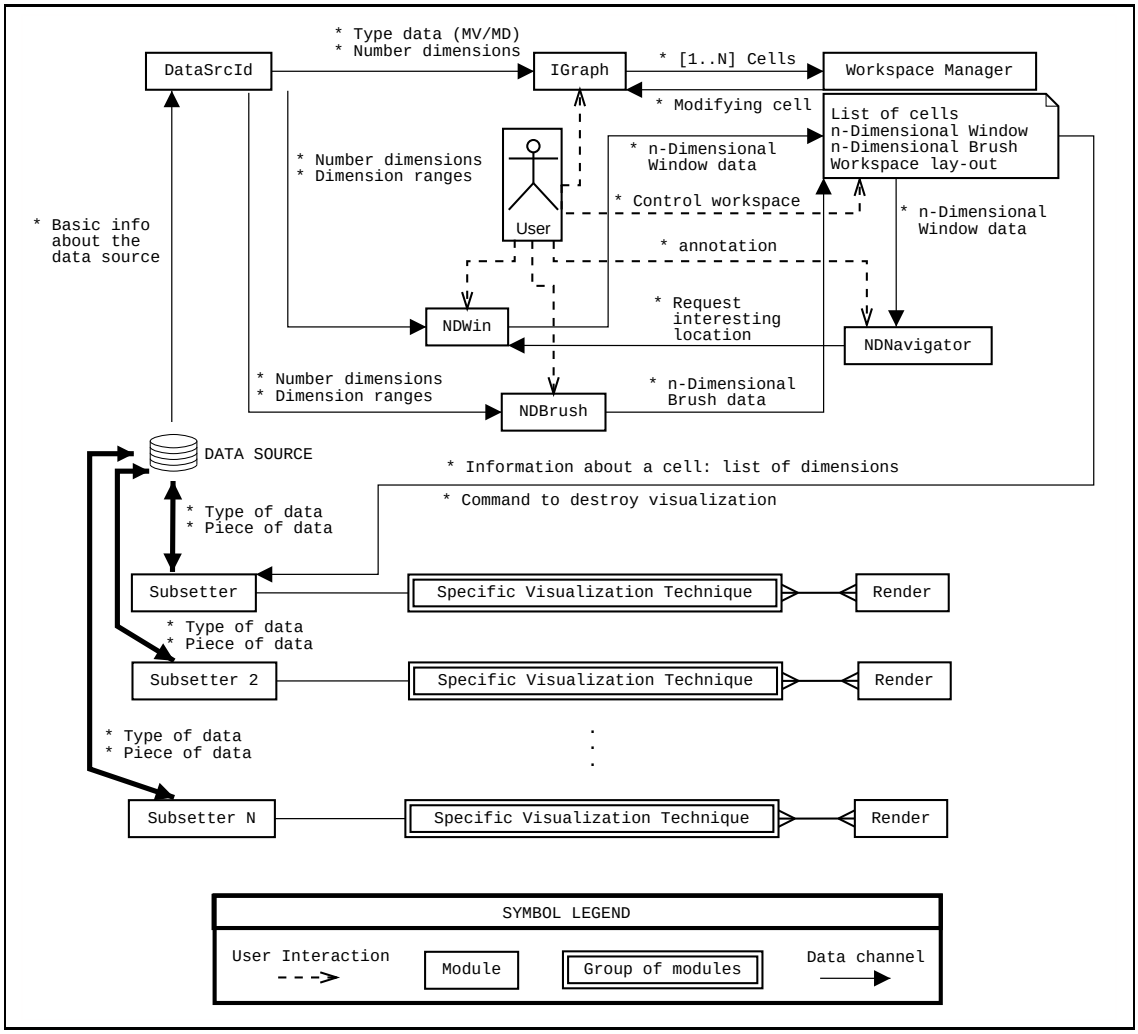


Figure 6.1: General structure of *HyperCell*. In this structure only one filter device is represented.

In Figure 6.1 the user is represented at the centre of the diagram, controlling the modules. The system initiates with four modules: DataSrcId, NDWin, IGraph, NDBrush (optional), and WSMan. These modules correspond to the creation of a single filter.

6.2.1 Describing the Data and Action Flow

The DataSrcId is the first module put to work, accessing the source of data and identifying the type (i.e. multivariate or multidimensional), the number of variables and their limits or range. This information is then passed downstream to the NDWin (to define the *n*-dimensional window of interest), IGraph (to let the user create the cells), and NDBrush (to define a *n*-dimensional brush object) modules; now these modules can reflect this information to the user via their respective GUIs.

The user then may change the parameters of the *n*-dimensional window of interest (i.e. the filter device), then send that initial configuration over to the WSMan (the workspace manager); the WSMan, in turn, registers the *n*-dimensional window information. Every modification made to the NDWin module and deliberately passed downstream by the user is stored within the WSMan and may be passed down to the NDNavigator (the module that stores the navigation actions).

The next natural step is to interact with the IGraph module and request the creation of one or more cells. This information is stored within the IGraph *and* within the WSMan. Once the creation of one or more cells reaches the WSMan the workspace manager displays the list of cells as elements of a scrolled list widget on the WSMan's GUI. At this point the WSMan may be set to simply show the list of cells on its GUI, or to immediately provide a visualization for the received cells. If the latter is the case then the WSMan module launches a Subsetter module (module that accesses the data source and extracts the subspace corresponding to a cell passed as argument) and a corresponding visualization for that type of cell. There are four types of visualization pre-stored within the WSMan that deal respectively with 1D, 2D, and 3D, and 3D cells with animation.

After the completion of this initial interaction, the user receives several visualizations corresponding to the requested cells, all linked together by the WSMan module. Changes made to the NDWin (e.g. modification of the limits of the window or changes to the focal point location) are passed down to all Subsetter modules associated with the WSMan, thereby affecting all the associated visualizations.

At this stage the user may start the exploration by (a) creating more cells or deleting the existing ones; (b) changing the parameters of the *n*-dimensional window, which characterizes a navigation operation; (c) interacting with the NDNavigator to annotate the current location or using it to re-visit previously stored ones; (d) interacting with the NDBrush to 'brush' a region of the *n*-space and observe the effect through the linked views; or, (e) creating another set of modules to simulate multiple filters.

6.3 Modular Visualization Environment

Basically we had two implementation options: (1) to program every aspect of the system – interface and visualizations – using standard programming languages such as C/C++ or Java, associated with graphical libraries such as OpenGL [177] and Java3D [180], software system for 3D computer graphics and visualization such as VTK [172] ; or, (2) to program within a modular visualization environment (MVE).

MVE is a visualization architecture in which several modules (blocks of code) are readily available for use through visual programming. These modules have a GUI that allows them to be 'wired' together in a 'visualization pipeline', using a mouse device (thus avoiding textual programming). The modules are usually classified in three broad categories, corresponding to the process stages of the Haber–McNabb dataflow reference model – Filter, Map, and Render (see Figure 3.3 in Chapter 3). Normally the MVE systems emphasize the mapping part of the pipeline by providing a great variety of techniques for visualization.

Although the first option offered the greatest flexibility, allowing total control over the GUI design and graphical representation, the MVE option seemed more attractive for several reasons:-

- By implementing the tools within an MVE platform, we gain access to the data analysis, visualization mapping and rendering facilities usually available in such system;
- MVEs are built upon the modular concept and most of the MVEs are based on the Haber–McNabb dataflow module, whose design is very similar to the framework we have adopted (c.f. discussion presented in Chapter 3);
- Several visualization algorithms, such as isosurfacing, scatterplot, and volume rendering, are already developed and available for use, thus avoiding the time-consuming task of coding them. Furthermore, creating alternative mappings for data (*parallel coordinates*, for instance) as a means to extend an MVE's functionality is not a difficult task;
- Most of the MVE systems offer flexible mechanisms to access data sources either as static data files or as modules that generate data on demand, and;
- Using an MVE platform is a good way of making this work available to the scientific community that already uses such systems.

6.3.1 The IRIS Explorer MVE System

HyperCell has been implemented in terms of the MVE system IRIS Explorer (IE) [203], as a set of new modules. By integrating into an existing environment (rather than building our own standalone tool), we immediately gain access to the rich functionality of that system.

The technical reasons for choosing IE as the MVE platform are:-

- Most of the IE modules are primarily designed to deal with numeric data, therefore ideal for the type of problem we are interested in.
- There exists an API library that allows the creation of custom-built modules. This was an essential feature once we had to code new modules to implement the functionality of *HyperCell.*
- It is possible to have a simulation module integrated with IE, creating the opportunity of computational steering.
- There exists a script language (Skm), an essential element to support creation of dynamic visualization for the cells.
- It is possible to take advantage of integral features like collaboration (e.g. COVISA [213]).

In addition, by choosing IE we immediately have access to support through the IRIS Explorer Centre of Excellence[1] based at the University of Leeds, where the work reported in this dissertation has been conducted.

6.3.2 Programming with IRIS Explorer

The implementation of some *HyperCell* features has proved a challenge, partly because of the strategy of developing a dynamic graphical interface within IE – this has entailed some additional effort in adapting to the constraints of the IE environment.

We could not rely only on the regular interface widgets available for creating a GUI for new modules. Most of the new modules had to have a dynamic interface, one that would change accordingly to the type of data (multivariate or multidimensional) and to the number of variables.

The solution we have found for this problem was to create the critical elements of the interface using a graphical library, namely OpenGL, and integrate them with the regular

[1] url:http://www.comp.leeds.ac.uk/iecoe/.

widgets of an interface, such as sliders and buttons. Therefore the new graphical elements of the interface are rendered within an OpenGL window embedded within the regular module's GUI.

The Module Builder is IE's auxiliary system for encapsulating user-written code as modules. A custom-built module can be implemented in C, C++ or Fortran languages. We have chosen to use C++ because of the possibility of organizing the code into classes. The access to creation and manipulation of IE's native data types is made available by a library of routines (API). The API's routines are also used to connect the 'central engine' of a module with the module ports. The module ports are the mechanism through which modules communicate to each other, passing information through the visualization pipeline.

6.4 Implementation Aspects of *HyperCell*

6.4.1 Support Classes

We have developed several support classes for the system and one of the most important classes is the *OglEnv* class. This an abstract class that provides the core functionality for the creation of a fully working OpenGL window embedded in a module's interface. The functionality of the *OglEnv* class includes providing an interface between the OpenGL rendering context and the native window manager system, and controlling the window's events, such as mouse clicks, re-paint request, etc. In order to keep compatibility with the IE system the *OglEnv* class has been coded to run in both Unix and Windows based environments.

For the modules to have an OpenGL based graphic interface they need to extend the *OglEnv* class by implementing their own drawing routine. Figure 6.2 shows the common class hierarchy followed by modules that have an OpenGL based interface (i.e. IGraph, NDWin, NDBrush, NDNavigator, and WSMan).

6.4.2 Communication of Information

Most of the communication between modules involves the creation and sending of data structures called *lattices*, or simple values called *parameters*. Lattices are data structures that represent multidimensional arrays of the following basic data types: byte, short, long, float, and double. The lattice data structure has two separate sub-structures, one to hold data values and the other to hold node coordinates. A node is a point in a lattice defined

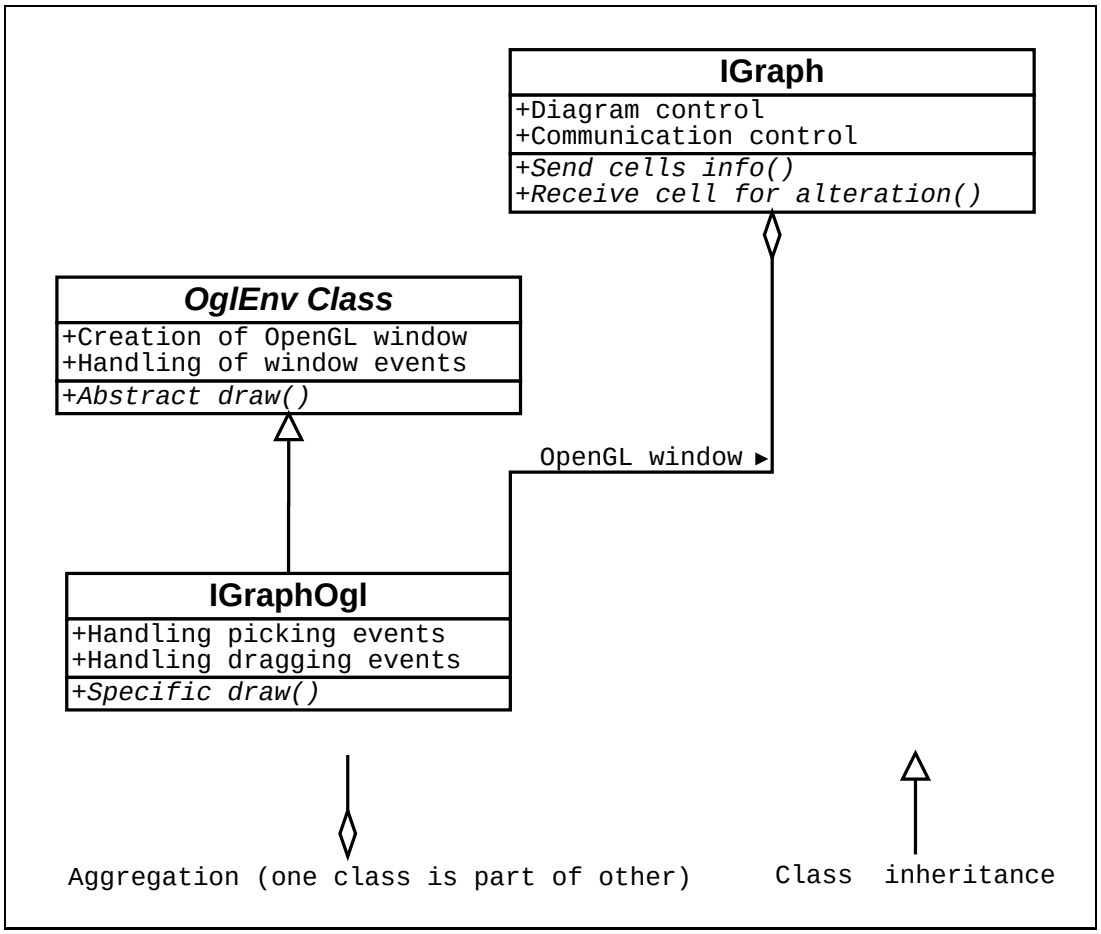


Figure 6.2: Example of relation between classes for the IGraph module.

by a unique coordinate or set of coordinates in Cartesian space, usually indicating the position of the data value (or values). By separating the structure into these two components – data and coordinates – a lattice gains flexibility, being able, for instance, to account for multidimensional data defined over irregular meshes. See for example Figure 6.3.

We have reserved the use of lattices to transmit the data to be visualized, and to communicate more complex parameter, especially those used to specify the complex elements, namely the *n-dimensional Window*, *n-dimensional Brush*, and the *Interaction Graph*.

6.4.3 Script Language

One of the useful aspects of *HyperCell* is the automatic creation of visualization for the cells. This aspect is realized with the help of Skm, the IE scripting language. The Skm scripting language is an alternative way of interacting with IE, replacing the high level interaction mechanism via visual programming. Through the use of a Skm script it is

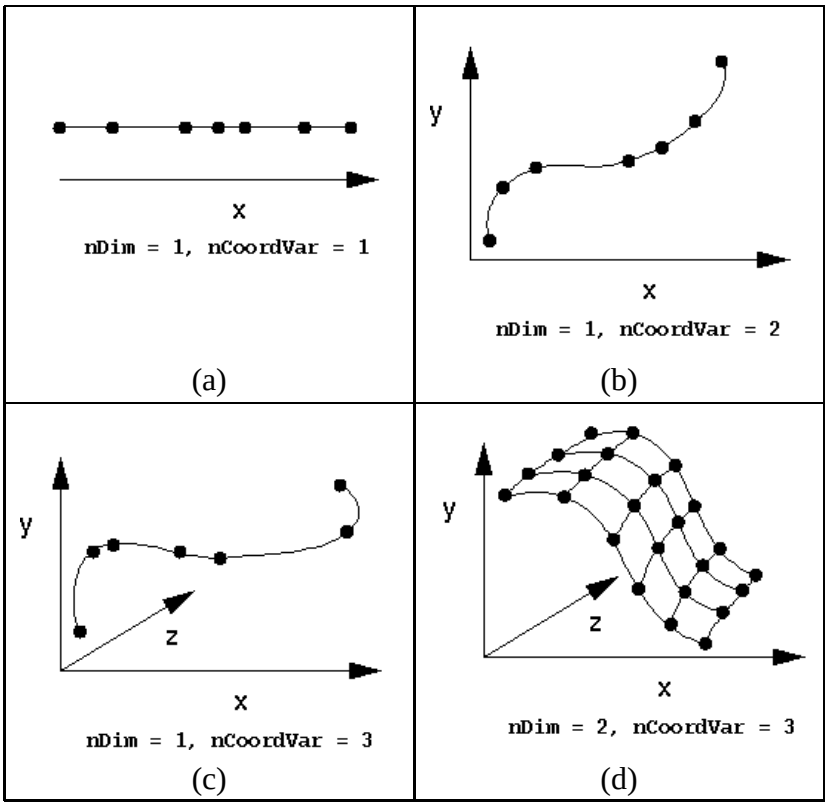


Figure 6.3: Example of various curvilinear lattices representing multidimensional data defined over irregular meshes. In pictures (a) through (c) we have 1D data defined over one-, two-, and three-dimensional coordinate spaces, respectively. Picture (d) shows 2D data defined over a three-dimensional coordinate space (pictures extracted from the IRIS Explorer user manual [127, Chapter 3]).

possible to have access to the same degree of functionality as using direct manipulation of modules to create a *map*[2].

The task of creating the pieces of map that generate a visualization for a specific cell is carried out by the WSMan module. The WSMan has been chosen for this task because this is a central module that receives all the information regarding the cells that have to be created. Consequently the *workspace manager* module has several pieces of hard-coded Skm scripts to create a basic visualization for a cell. There is a different piece of code for each different dimensionality and for each different type of data (multivariate or multidimensional)[3].

Note that the automatically generated pieces of map to visualize a cell is only a 'suggestion' made by the WSMan. Once a cell has been created and visualized, the user is

[2] A map is the IE term for a visualization pipeline.

[3] Ideally these pieces of maps should be provided to *HyperCell* via Skm source files which the user can supply, rather than hard-coded. However we have hard-coded them just as a fast way of getting the system up and running in the shortest time.

free to create his/her own visualization, because a cell is nothing more than a lattice of data.

6.4.4 The 'Chameleon' Module

The interfaces of the Subsetter modules change depending whether the data type is either multivariate and multidimensional, as shown in Figure 6.4.

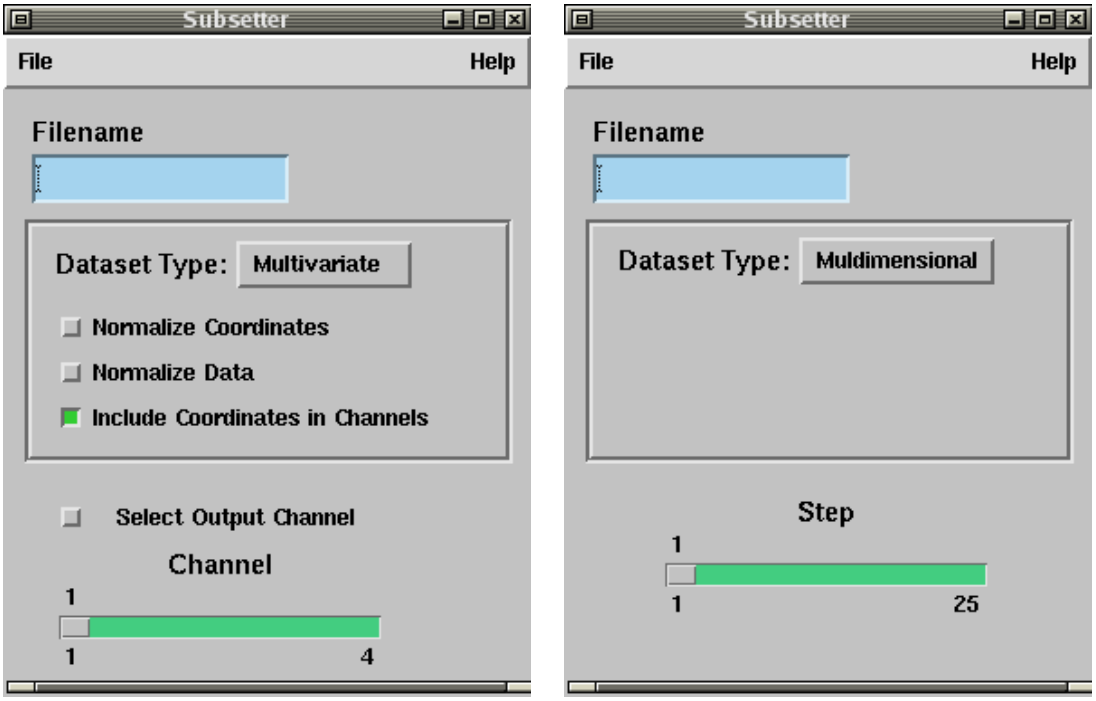


Figure 6.4: Different GUIs for the Subsetter module, depending whether the data type is either multidimensional (a), or multivariate (b).

The picture on the left of Figure 6.4 shows the Subsetter interface for multivariate data. The Subsetter in this mode allows the user to 'normalize' – i.e. map a variate range into the range $[0,1]$ – both coordinates and data values. The 'normalization' of variables is important for the cases where the variables are in different scales, therefore mapping their values directly to coordinates on the *visual space* would create coordinate systems with different scales on the axes. (This feature is later demonstrated in Figure 7.12 of Chapter 7, Section 7.3.2.2.) Figure 6.4-(b) has a slider called 'Step' at the bottom of the GUI that appears on the interface only if a 4D cell (i.e. 3D cell + time dimension) is to be extracted. This slider enables the user to control which 'time frame' of a 4D cell to send downstream for the mapping part of the pipeline.

The Subsetter also permits the inclusion of the variates that have not been assigned to the coordinates of the *visual space* as 'channels' of the output lattice. Remember that in the multivariate case the variates assigned to form a cell are mapped to a coordinate system on the visual display – thus individual data items are mapped to Visual Marks located within the *visual space*, but the remaining variates can still be assigned to *graphical properties* of the Visual Marks.

6.4.5 Access to Data

The Subsetter module stores multivariate data differently from multidimensional data. All multivariate datasets are stored as one-dimensional lattices with as many data values per point as the original number of variates. In this case the number of data points corresponds to the number of observations. Multidimensional data, however, is stored as multidimensional lattices where the number of data dimensions in the lattice correspond to the number of dimensions in the original data source.

The extraction of subspaces is currently performed after the whole data is read into memory and represented as a multidimensional lattice. Then a recursive routine extracts the 'pieces' of lattices corresponding to the requested subspaces and converts them into new one-, two-, or three-dimensional lattices.

The access to individual data elements, during the extraction process, is done through a *pointer* to the part of a lattice that contains the data. The access to data in memory via pointers makes it relatively straightforward to adapt the extraction routine to deal directly with data files stored in disks. This is possible mainly because the access to data from a file in C++ is done almost in the same way as the access to data in memory – using a pointer (in fact it is a *stream* object that encapsulates a pointer). Therefore the use of a pointer to retrieve data in files allows random access to bits of data on disk, instead of the more restrictive sequential access.

Making the Subsetter module extract sub-lattices directly from disk may slow down the overall performance of the system, as a result of the user requesting the creation of various cells. On the other hand, this approach allows *HyperCell* to be virtually free from any limits in terms of number of variables and number of observations in a dataset file.

The overall difference in terms of performance between (a) a Subsetter that extracts the sub-lattices from a multidimensional data source previously read from file and stored in the main memory; and, (b) a Subsetter that leaves the data on disk and reads in only the sub-lattice; will consist of the expected overhead between retrieving data from the secondary memory and retrieving data from the main memory plus the time to skip any

data offset and access the beginning of the required piece of data. The computation time used to calculate 'where' the bits of data are located within the original data source is the same for cases (a) and (b).

6.4.6 A Final Example

Figure 6.5 depicts the initial map shown after the initialization of the *HyperCell* system. This representation is the same, regardless of the data type.

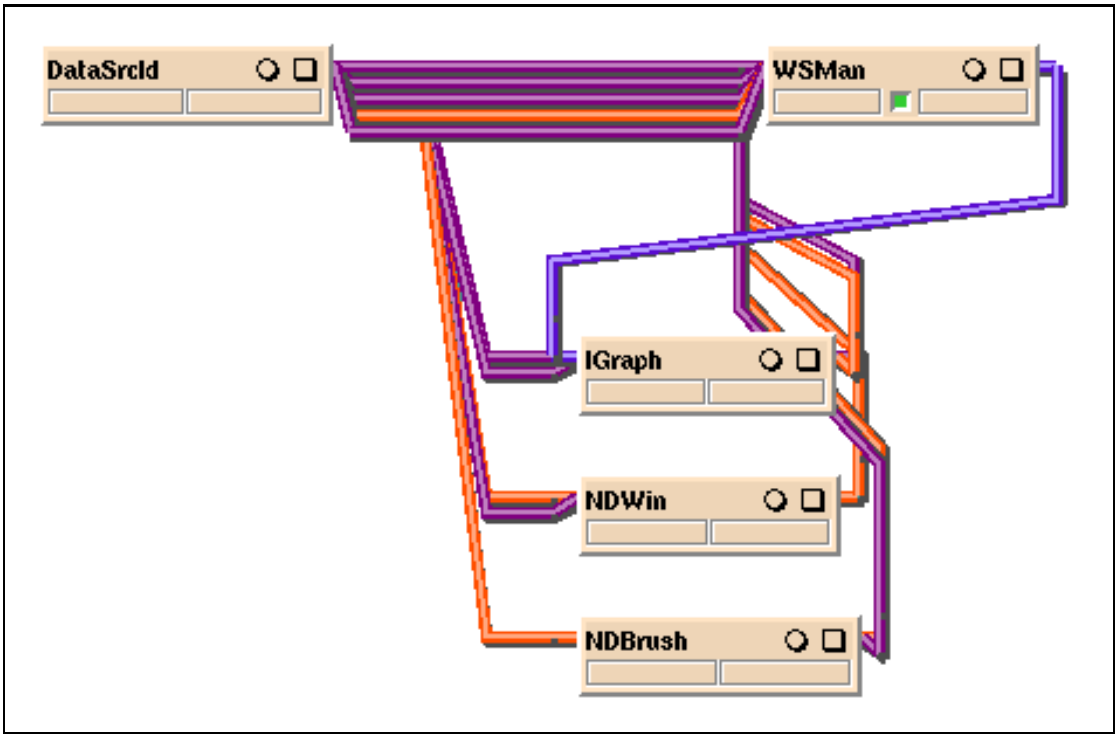


Figure 6.5: Initial IE map representing the *HyperCell* method.

Figure 6.6 shows the same general structure depicted in Figure 6.1, but this time as an IE map. Note that in that map two cells have been created.

Here is a list of some positive aspects related to the implementation:-

- **Access to Data:** The largest multivariate dataset used during the appraisal phase had 60 variates and the largest multidimensional dataset had 6 dimensions with 20 data points in each dimension (i.e. a grid with 20^6 data points). There is, however, no empirical evidence as to the size of dataset at which performance of the system begins to degrade.

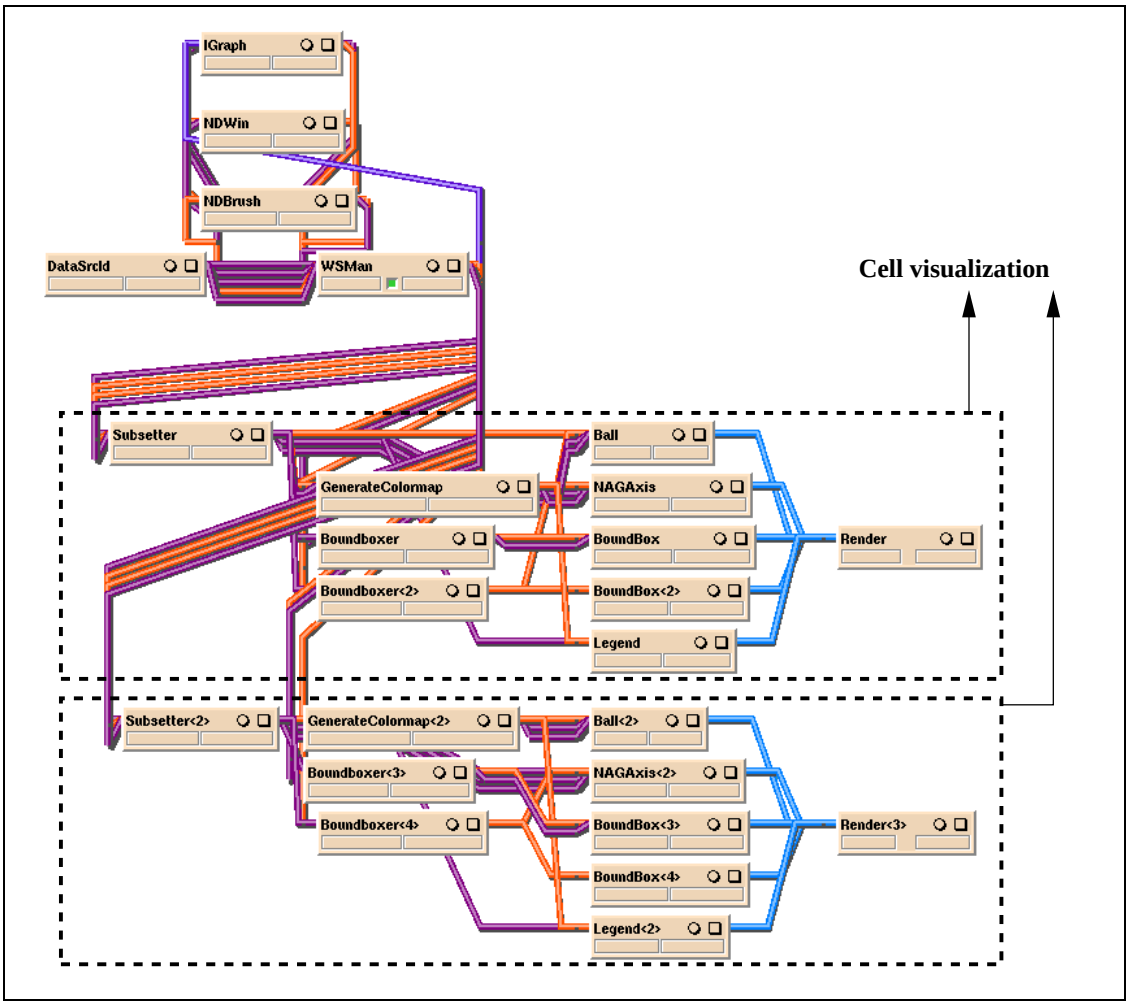


Figure 6.6: IE map representing the *HyperCell* method after two cells have been created.

The practical experiments have lead us to believe that the number of dimensions of a dataset is not a strong constraint in terms of memory and processing power because each individual cell demands only a small amount of memory and rendering power to be displayed. Furthermore, the user is in control of the process of creating cells, consequently only the 'filtered' data needs to be present in the memory, while the rest of the data remains, say, in disk storage.

- **Interface scalability:** The interface seems to scale with the number of dimensions, as shown in Figure 6.7-(a) for the *n-dimensional Window* diagram. In that image the vertices of the diagrams are drawn a little off the original circle when the number of dimensions is greater than 15. This simple drawing strategy avoids the overlapping of vertices when the number of dimensions is large, thus making all of the vertices visible and 'clickable'. The inclusion of this feature into *HyperCell* has been mo-

tivated by one of the case studies (the multivariate astronomy data) presented in Section 7.3.2 of the next chapter, in which we have a total of 57 variates.

- **Dynamic interface:** The implementation of a graphical user interface using an OpenGL drawing area greatly increases the control over the behaviour of the interface. This aspect was fundamental in ensuring that the designs proposed in Chapters 3 and 4 were met.

- **Use of simple visualizations:** *HyperCell* has not introduced a new visual representation beside those of the tools' GUI. Rather, it takes advantage of the users' cognitive experience in visualization, by employing standard 3D visual algorithms such as isosurfacing, line graphs, and volume rendering.

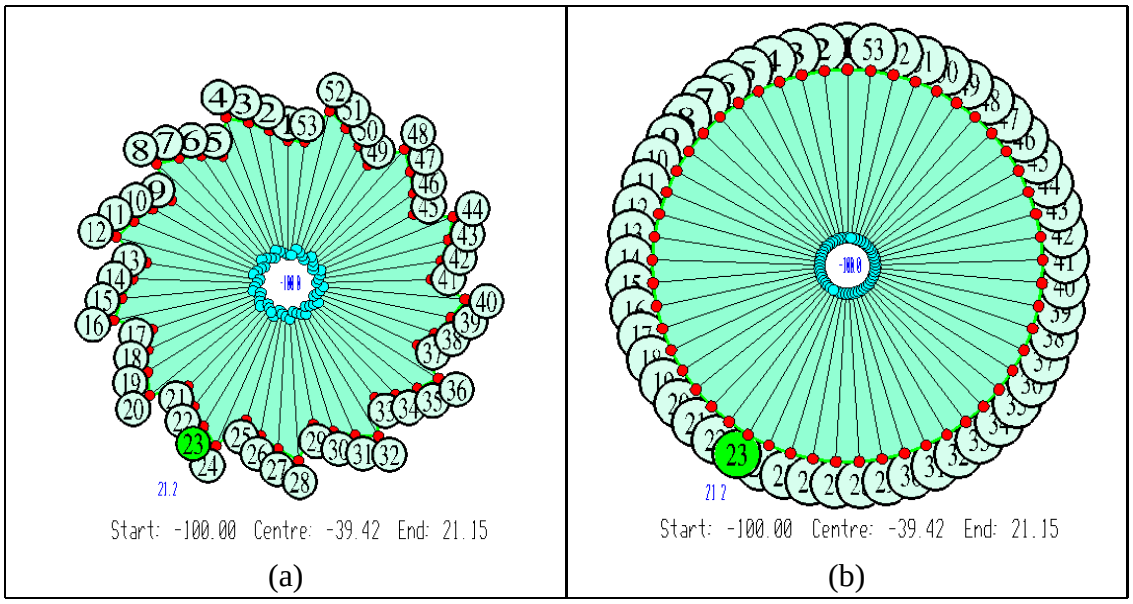


Figure 6.7: The *n-dimensional Window* diagram scales well to a high number of dimensions (total of 53 dimensions), as shown in picture (a). Picture (b), however, has not been adapted, showing that as the number of dimensions increases it becomes difficult to see each vertex because of occlusion. Note that clicking on a vertex in both cases brings that vertex and the corresponding spoke to the front of the diagram (e.g. vertex 22 on the diagram).

6.5 Summary

In this chapter we have presented the general structure of *HyperCell*, showing how the elements introduced in the previous two chapters come together to form a modular system. We have also discussed issues related to the *HyperCell* implementation.

Early on in the development of this work a decision was taken towards the use of a modular environment system instead of programming every aspect through graphical libraries and programming languages. Implementing the system in terms of the IRIS Explorer environment has several benefits: (a) we gain access to the great functionality available in such a system; (b) we build upon a modular concept compatible with the Haber–McNabb dataflow model, which in turn conforms with the suggested reference model of Chapter 3; (c) we have access to various visualization algorithms already implemented and tested, thus speeding up the implementation phase; (d) we can count on flexible mechanisms to access data sources; and, (e) we broaden the reach of our system by using a platform that has been increasingly used by the scientific community that needs visualization.

We have also shown the method used to implement the various components of *HyperCell* in terms of IRIS Explorer modules. This includes the creation of support C++ classes, the use of IE's API interface, creating pieces of map with Skm scripting language, and implementing interface elements using the OpenGL library.

Finally we have presented an example in which the *HyperCell* system is realized in terms of a IE map.

Chapter 7

Appraisal

THE EVALUATION METHODOLOGY used to appraise the *HyperCell* visualization technique is the central subject of this chapter.

The evaluation procedure involves basically two phases: 1) a *design evaluation* phase, in which the various components of the *HyperCell* technique are assessed according to guidelines related to the use of multiple coordinate views, the level of coordination of a multiple views environment, and visualization issues; and, 2) a *testing* phase, in which the proposed visualization technique has been applied to several case studies involving real world-based high-dimensional applications.

The last part of this chapter contains some remarks and conclusions drawn on the overall results obtained from the experience of applying *HyperCell* to the case studies.

7.1 The Appraisal Strategy

The evaluation of visualization methods has been increasingly recognized as an important aspect in supporting visualization research. Wong and Bergeron in [212], and Grinstein *et al.* in [83] have recognized the importance of directing visualization research towards formal and rigorous evaluation methodologies. More recently Grinstein, Trutschl and Cvek in [86] have proposed a metric for comparison of visualization techniques, while

Tory and Möller in [193] have suggested several approaches based on human factors paradigms that might be useful in evaluating visualizations.

All these authors agree that visualization research will certainly benefit from applying the knowledge and experience from perception and cognition sciences, and human factors research to the design of visualization methods. Of particular interest is the work of Tory and Möller – we have found one of their suggested approaches for evaluating visualization very promising for our work. Therefore in this thesis we have followed an adaptation of their evaluation methodology presented in [193], which comprises two phases:-

- **Design evaluation**: In this phase we perform a qualitative evaluation of the *HyperCell* design in terms of a set of guidelines regarding perception, cognition, and human factors.

- **Case studies**: This phase comprises applying the system to real life problems from different backgrounds, involving both multidimensional and multivariate data in situations that represent the investigation scenarios described early on in Chapter 5 (c.f. Section 5.3).

We believe that the qualitative evaluation of the visualization results together with the user feedback from these case studies offer the evidence necessary to ascertain whether *HyperCell* has accomplished the objective of proving a uniform treatment for high-dimensional data that conveys some insight.

We have decided not to conduct any user studies because they seemed not appropriate for our work at this stage. There are two reasons for this. Firstly, the intended audience is not clearly identified due to the generality of the subject being studied. This makes it difficult to find a significant number of subjects that could take part in a controlled experiment. However, in our case studies we have been in contact with three potential users, who have presented us with real high-dimensional applications that needed visualization. Secondly, user studies are very effective for identifying problems with interface and user interaction, but they usually are not as good in highlighting potential problems with, or benefits of, visualization ideas [201, Chapter 2]. By contrast we think that the application to practical problems, followed by the user feedback on the results, provide the elements necessary to have an overall evaluation of the *HyperCell* method.

7.2 Phase 1: Design Evaluation

7.2.1 Guidelines for Using Multiple Views

The first set of guidelines we consider have been proposed by Baldonado *et al.* [204] to aid the design of multiple views systems. A multiple views system, by their definition, "uses two or more views to support the investigation of a single conceptual entity."

The first four guidelines are intended to guide *when* to use multiple views, whereas the last four guidelines provide advice on *how* to choose between the several types of view presentations and interactions.

1. "Rule of Diversity: *Use multiple views when there is a diversity of attributes, models, user profiles, levels of abstraction, or genres.*"

   In *HyperCell*'s case this rule applies because there is a diversity of attributes represented by the several variables of a high-dimensional dataset.

2. "Rule of Complementarity: *Use multiple views when different views bring out correlations and/or disparities.*"

   One of the basic objectives of *HyperCell* is to help bring out any meaningful correlation or interesting behaviour between variables.

3. "Rule of Decomposition: *Partition complex data into multiple views to create manageable chunks and to provide insight into the interaction among different dimensions.*"

   The *HyperCell* relies on the presentation of high-dimensional data as multiple cells as result of the application of one or more filters to the data.

4. "Rule of Parsimony: *Use multiple views minimally.*"

   HyperCell avoids the automatic creation and simultaneous display of all possible subspaces obtained by combining variables. We believe this would overload the user with information that might be not required to the task at hand. Instead *HyperCell* delegates the responsibility of creating the subspaces to the user.

5. "Rule of Space/Time Resource Optimization: *Balance the spatial and temporal costs of presenting multiple views with the spatial and temporal benefits of using the views.*"

   This rule draws attention to the fine balance between showing all views simultaneously to allow comparison and the cost of generating them and the screen space

needed to present them. This is one of the reasons we have proposed two different lay-outs of cells – the 2D 'fruit machine' and the building lay-out – as explained in Chapter 5, Section 5.4.1. Further discussion on this issue is presented in Section 7.4, where we present the overall results from applying *HyperCell* to the case studies.

6. "Rule of Self-Evidence: *Use perceptual cues to make relationships among multiple views more apparent to the user.*"

   HyperCell follows this rule through the use of *linking and brushing*, defined by Baldonado *et al.* as *coupled interaction*. Other forms of perceptual cues can be created by arranging the multiple views so that similar axes are aligned.

7. "Rule of Consistency: *Make the interfaces for multiple views consistent, and make the states of multiple views consistent.*"

   HyperCell uses the same visual representation for cells with same dimensionality, regardless of the combination of variables. Also we have tried to keep consistency between the representation of variables in the interfaces of IGraph, NDWin, and NDBrush modules. The advantage of using a consistent visual representation for the data is evident in remarks of the case study #3, described in Section 7.3.3.

8. "Rule of Attention Management: *Use perceptual techniques to focus the user's attention on the right view at the right time.*"

   One of the most important events in *HyperCell* is the creation of a cell and whenever this happens the last window to pop up is the one that contains the rendering of the cell.

The first four guidelines were fundamental in improving an earlier design of *HyperCell* which had been implemented for Microsoft Windows environment as a bespoke C++ program using OpenGL (see Figure 7.1). That early version had only two windows: one on the left-hand side containing a tree representation of cells (leaves) and workspaces (roots) created; and, the visualization window on the right-hand side in which the IGraph interface and the visualization of the cells are presented.

The rule of *diversity* helped us to see the need to assign separate views for the interface and for the visualizations produced, since they convey distinct informations. The *decomposition* rule has lead us to the current *HyperCell* design in which a workspace is used to accommodate several cells linked to the same region of interest around a focus point, in contrast with the early version whose function was simply to organize the

cells into groups. We felt that by associating location in *n*-space with a workspace it became easier for the users to understand that cells contained in a workspace correspond to 'slices' taken from the same location in *n*-space. This association, in turn, helps the partition of the high-dimensional data space into "manageable chunks", as suggested by the *decomposition* rule.

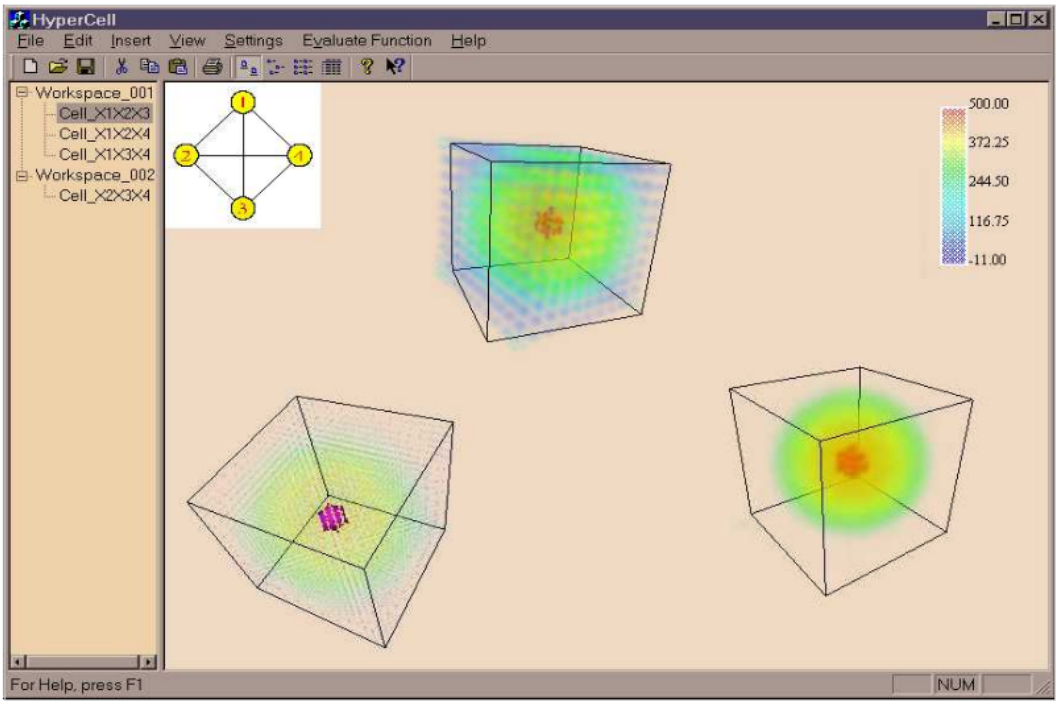


Figure 7.1: Screenshot of an early version of HyperCell.

Therefore the latest *HyperCell* version is based on a more extensive use of multiple views, supported by the first four guidelines. This version has also tried to follow the last four guidelines (as discussed when the guidelines were introduced on page 130). However, it seems to comply minimally with the *self-evidence* and *attention management* rules.

Regarding the *self-evidence* rule we believe the *linking and brushing* to be very simple for two reasons: 1) it does not allow the user to create the brushing in the *visual space* (i.e. the rendering window) but only via *data space* (i.e. the NBrush interface); and, 2) the brushing mechanism only shows a box surrounding the brushed data, not highlighting them nor eliminating the non-brushed data.

With respect to the rule of *attention management* we felt that making a created cell "pop up" was a simple feature to satisfy this rule, but that other mechanisms should also be provided. We discuss such mechanism later on in Section 7.4.

7.2.2 Classifying the Level of Multiple Views Coordination

North and Shneiderman [150] introduced a four level classification scale for multiple coordinate views, according to the degree of flexibility in *data* (different datasets can be loaded into the visualization); *views* (different sets of visualizations can be selected as appropriate for the data); and, *coordination* (different types of coordination between several views can be chosen by users according to their intent in exploring or navigating the relationships in the data).

- *Level 0:* The lowest level of flexibility. A file manager is a typical example of this level. The data is always the same (file system of a selected hard disk), the views are always the same (one view shows a tree structure corresponding to the directory structure while another shows the content of the directory selected in the first view), and the coordination is the same (clicking on a directory on the tree structure shows the content of that directory on the other view).

- *Level 1:* The selection of data to be visualized is flexible but the same does not apply for views and coordination. For example a *Treemap* [175] can represent any hierarchical data but the pair of views the *Treemap* provides (i.e. the hierarchical organization and the detail window) remains the same. Most visualization tools are in this level.

- *Level 2:* This level covers systems with flexible selection of data and views, but the coordination is still fixed. The *Spotfire* system [3] is a representative example of this level. *Spotfire* can represent multivariate data using several windows, each one of them showing different views such as scatterplots or bar charts; the views provided by this system allow the *brushing and linking* style of coordination – i.e. corresponding data items selected in one view appears selected in the other views.

- *Level 3:* The most flexible of all levels. Examples of systems in this level are the *Spreadsheet Visualization* [44] – which arranges several small 3D views in a matrix like 2D grid allowing users to select a whole row or column of views to synchronize their 3D navigation; and modular visualization environments which provide three levels of flexibility: data (several data formats are acceptable), views (several different views can be custom-made to satisfy the user), and coordination (the views may be dynamically linked).

The early version of *HyperCell* (see Figure 7.1) is classified as *Level 1* because although various functions' datasets could be loaded only one type of 3D visualization was

provided. The low level of coordination achieved lead to a review of the platform adopted, and a search for new flexible options.

In contrast the present version has been designed with the *level 3* characteristics of an MVE in mind. Any type of data, whether multivariate or multidimensional, can be read, assuming they are stored in either IRIS Explorer lattice or in *XmdvTool* formats; several views can be applied to the cells, e.g. scatterplots, isosurfacing, line graph, etc.; and, the coordination is also flexible, i.e. cells can be coordinated together if they belong to the same workspace, and multiple workspaces can be created, thus allowing different coordinations.

7.2.3 Visualization Specific Guidelines

Here we introduce the last set of guidelines, aimed to help the elaboration of visualization. These guidelines have been compiled by Nigay and Vernier in [147][1].

- Visualization is domain and task dependent, therefore a design must either 1) be domain and task specific; or, 2) implement flexible domain-independent subtasks such as overview, zoom, filter, details-on-demand, relate, history, and extract (these subtasks were introduced early on in the TTT taxonomy by Shneiderman [176], presented in Chapter 2, Section 2.2.3).

  HyperCell provides at least one alternative to each of the domain-independent subtasks proposed by TTT taxonomy: overview – the lay-out of cells; zoom – performed in the *visual space* (also available in the *data space* if a module to generate the data anywhere is present); filter – only data within a *n*-dimensional window can be visualized; details-on-demand – in the multivariate case clicking on a data item in a view reveals all the variate values of that data item; relate – *brushing and linking*; history – the NDNavigator module allows users to go back to previously stored locations in *n*-space; and, extract – this corresponds to filtering data items or variables to form a cell.

- To support users with different tasks and requirements, multiple visual representations of the data should be available. This involves:-

  1. *Changing representation in views should be easy. HyperCell* allows users to 'plug in' their own representation for a cell, since *HyperCell* has been implemented within a MVE system.

[1] In fact the list of guidelines has been taken from Tory and M¨oller [193], who presented them in a more compact form.

2. *Using multiple views is not always appropriate.* It is at the user's discretion to use several views to explore the *n*-space or just one dynamic view (c.f. Dynamic Cell Chapter 4, Section 4.4.1).

3. *Continuity should be maintained so the user does not get lost when switching between representations.* We have tried to follow this guideline when applying the Dynamic Cell mechanism and the results obtained with the case studies indicate that the user can keep a sense of continuity when moving between different subspaces (see Figures 4.5 and 4.5 in Section 4.4.1 to see how continuity between cells has been achieved).

- The following variables should always be visible:-

1. *The set of data elements (an overview).* Showing all data items simultaneously is the main challenge for high-dimensional visualization techniques. Several levels of overview are possible, however, by using one of the cell lay-out strategies to display, for example, all possible 2D cells.

2. *Relationships between data elements*. This is evident only through the use of *brushing and linking.*

3. *Methods of locomotion*. This task is performed by the NDWin module when the user changes the *n*-dimensional window parameters and this corresponding modification is stored within by the NDNavigator module.

4. *Navigation history*. This is done by the NDNavigator module.

- *Data at the focus of interaction should be undistorted and represented at the highest possible resolution.* In our case the user controls the degree of resolution for each view by either setting a parameter on the Subsetter's GUI or adding a sampling module to the visualization pipeline of a cell.

- *Navigation tools should be reused to maintain consistent interaction metaphors throughout the system.* We have used the same metaphor representation for the diagrams used by IGraph, NDWin, and NDBrush modules, and a somewhat similar metaphor for the NDNavigator module.

Based on the above guidelines we can say that *HyperCell* has had some problems in making evident *relationships between data elements* (see the earlier comment on *linking and brushing*, at the end of Section7.2.1), and providing *an overview* of the whole dataset.

Of course, the latter is exactly the main problem of this research which is to promote the visualization of a high-dimensional data space. However we have been satisfied with the results obtained with the 'fruit machine' lay-out, as later discussed in Section 7.4.

7.3 Phase 2: Case Studies

In this section we investigate three different sets of high-dimensional data involving both SciVis and InfoVis applications. They are: the study of the Rosenbrock function and the trajectory generated by an iterative optimization algorithm; an astronomy dataset corresponding to the classification of sources of light in the sky; and data from a course management system used to aid instructors controlling a distance learning course.

7.3.1 Case Study #1: Multidimensional Rosenbrock Function

In the first case study we use *HyperCell* to explore a well known function of four variables from the optimization world, the chained Rosenbrock function [168]. This is a generalization of the original Rosenbrock function (the case $n = 2$), which has an interesting banana-shaped valley – the shape of the function is shown in Figure 7.2. The four dimensional generalization however is much harder to envisage.

The function is defined by the following expression:

$$F(x) = \sum_{i=2}^{n} [100(x_{i-1}^2 - x_i)^2 + (x_{i-1} - 1)^2] \tag{7.1}$$

We choose a four dimensional example for ease of presentation, but the idea scales up to higher dimensions. Indeed in the concluding part of this case study we extend to the six dimensional case.

It is easy to determine the *minimum* point of $(1,1,1,1)$, almost by inspection, with corresponding minimum value of zero – but what is the behaviour of the function near the minimum? This is the sort of sensitivity analysis question that is increasingly important in optimization problems.

Let us suppose for a moment that this is an unknown function and the user has no idea about the function features and properties. This is characteristic of the *Scenario 4* of the exploratory scenarios described early on in Section 5.3 of Chapter 5. In this particular scenario the user has no idea of *what* to look for, nor *where* to find anything interesting during exploration. The recommended exploration strategy for this scenario is: "to use disjoint four-dimensional cells to cover a larger area in a short time, until an interesting

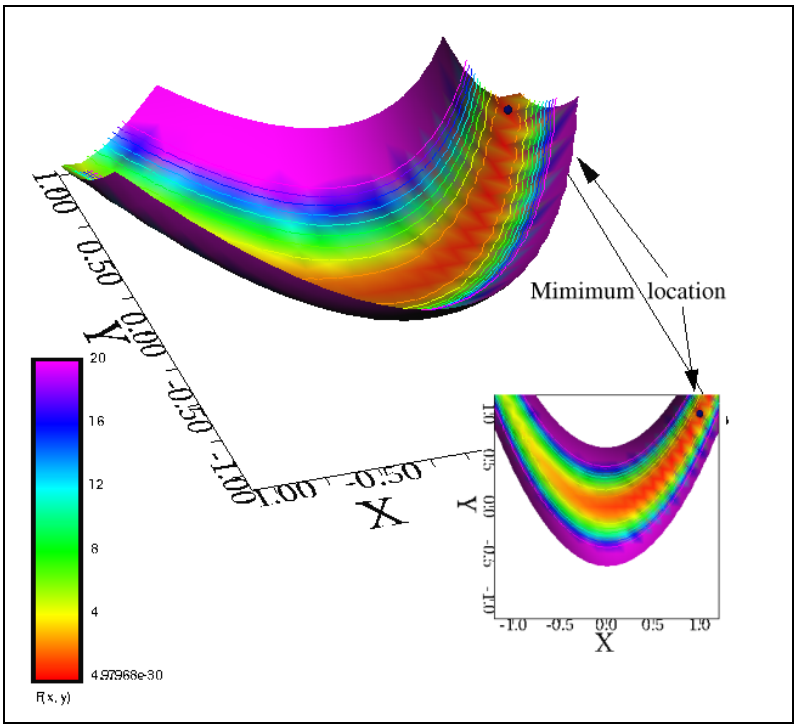


Figure 7.2: The main picture shows a 3D view of the Rosenbrock's famous Banana-shaped Valley having its height and colour associated with the function value. The small picture on the bottom-right corner is a top view visualization of the same function. The minimum location is depicted by a black dot indicated by the arrows.

location is found; then generate several cells at that location to explore the region." In this particular case a single four-dimensional cell is sufficient, because such a cell involves all dimensions. Figure 7.3 shows a sequence of images corresponding to twelve snapshots taken from the animation corresponding to the four-dimensional subspace *cell-(1,2,3,T:4)*.

In Figure 7.3 *dimension 4* is used as time, therefore in each 'time frame' of the animation the focus point is moved along *dimension 4*.

At the end of the inspection of *cell-(1,2,3,T:4)* we verified (by looking at the shape of the isosurface of value near zero) that when the focus point is at $(X_1, X_2, X_3, +1.0)$ the function reaches the lowest value at $(1,1,1,1)$. To understand the behaviour near $(1,1,1,1)$, we can repeat the same process using another four-dimensional cell, for example *cell-(2,3,4,T:1)*, which would produce a similar result. Therefore the use of this exploration procedure has allowed us to confirm an interesting location.

According to the recommendation for *Scenario 4*, once an interesting location in *n*-space has been found through the use of a four-dimensional cell we may proceed to explore this region by, say, generating several cells at this location. This is shown in Figure 7.4, in which we probe in each coordinate direction through all possible one-

dimensional cells generated at the interesting location. Note that the lowest function value is at coordinate +1.0 for all dimensions, and that the function does not seem to be very 'sensitive' around that location (i.e. small continuous changes of the function value as we move away from the coordinate +1.0).

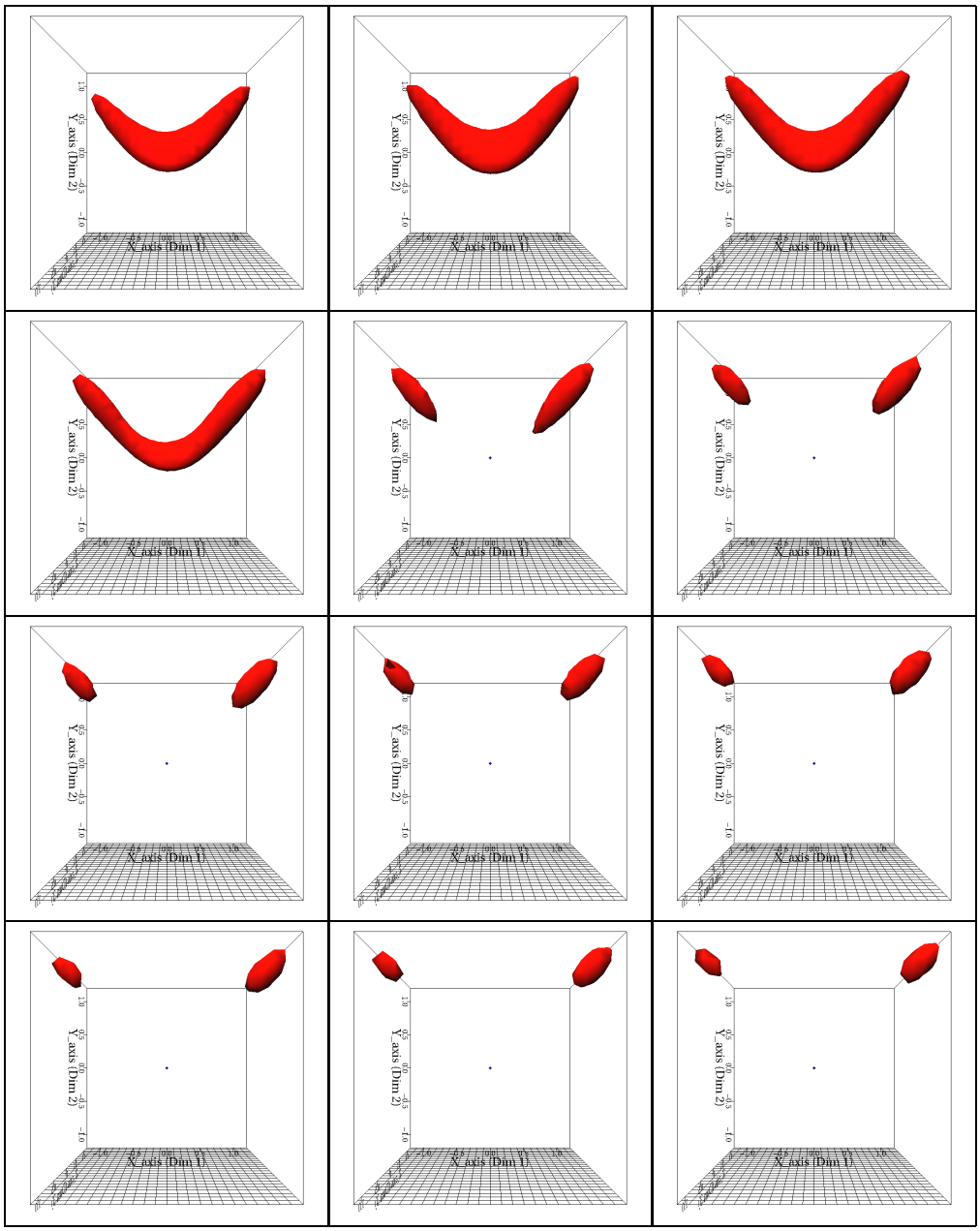


Figure 7.3: Visualizing the four-dimensional subspace *cell-(1,2,3,T:4)* with isosurface value set to +10.0. This figure contains a sequence of images taken from the animation at different time frames (the sequence is considered left to right, top to bottom). The pictures correspond respectively to the *cell-(1,2,3)* with focus point at $(X_1, X_2, X_3, \hat{X_4})$, where $\hat{X_4} \in \{-0.1, 0.0, +0.1, +0.2, +0.3, +0.4, +0.5, +0.6, +0.7, +0.8, +0.9, +1.0\}$.

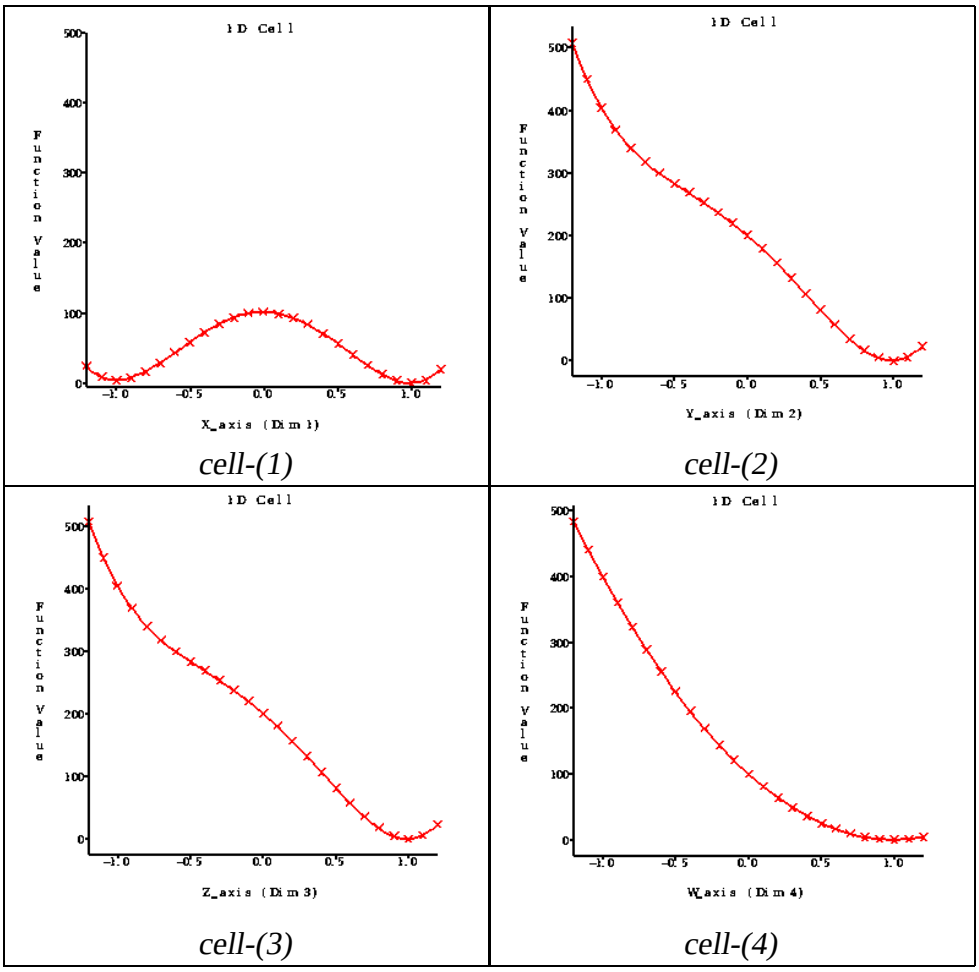


Figure 7.4: Visualizing the four-dimensional Rosenbrock function through an array of 1D cells at the minimum, i.e. $(1,1,1,1)$.

Alternatively the Dynamic Cell mechanism (c.f. Chapter 4, Section 4.4.1) could be used to explore progressively subspaces of different dimensionality with a smooth transition between these subspaces. This is shown in the following sequence of pictures, aimed to study the behaviour of the 4D function near the minimum point $(1,1,1,1)$. The corresponding visualizations are shown in Figure 7.5.

The top left image shows a graph with *dimension-1* allowed to vary over its range – this is *cell-(1)* visualized as a 1D line graph; next, top right image, we allow *dimension-2* to vary and display as a contour plot – this is *cell-(1,2)* visualized as a 2D coloured map with colours assigned according to the function values. A dashed line on the top right image through the focus point shows where the *cell-(1)* of the top left image fits (follow arrow in black between the top images). Notice in this cell (*cell-(1,2)*) the low values in the neighbourhood of $(-1.0,1.0)$ and $(1.0,1.0)$ in these two dimensions, with

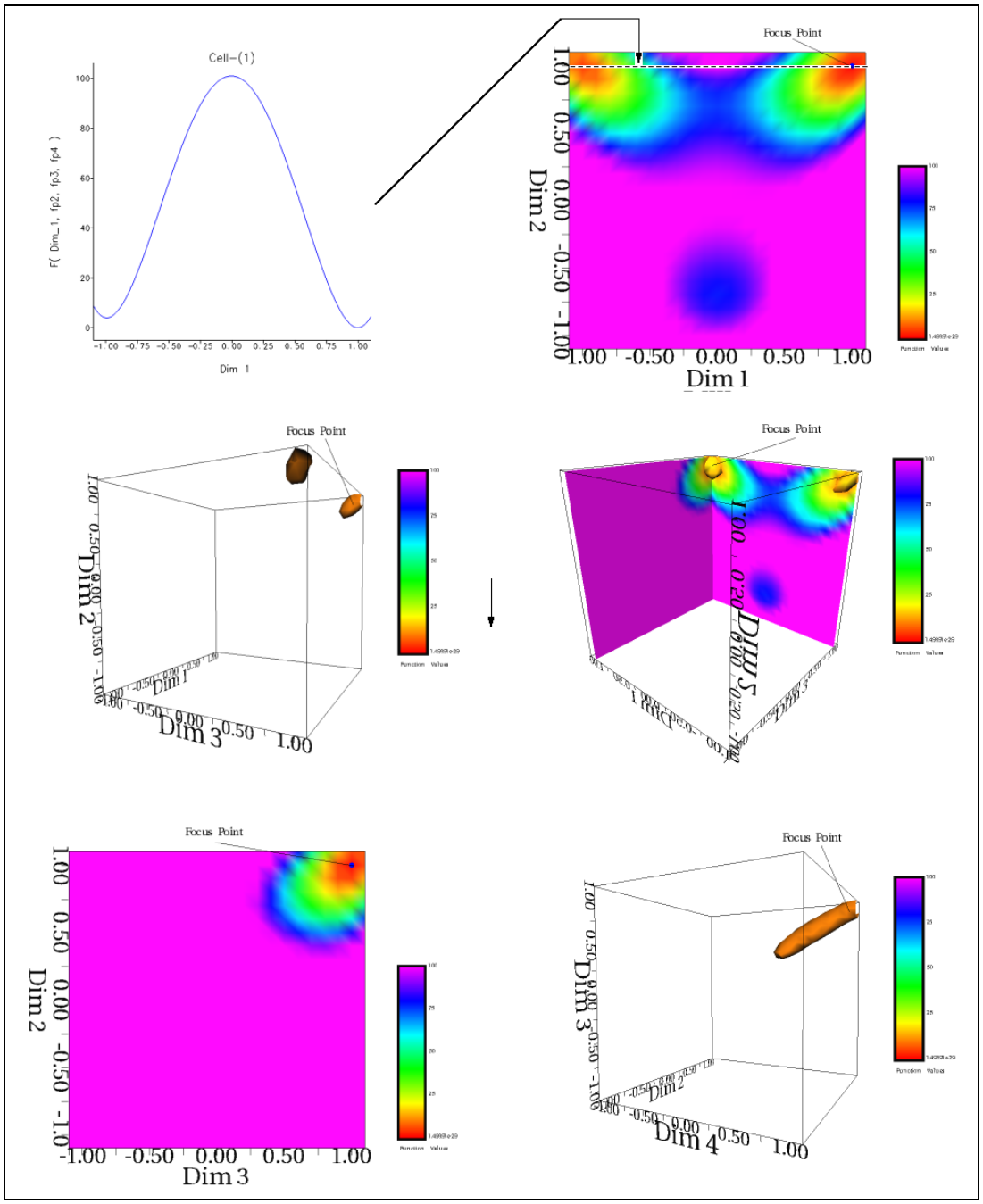


Figure 7.5: Progressive exploration of the Rosenbrock function in 4D. These pictures are visualizations with different dimensionality and combination of dimension, taken from the same focus point located at $(1,1,1,1)$.

an indication also of a low area around $(0.0,-0.6)$. (Dimensions 3 and 4 are fixed at their focus values, namely 1.0.)

In the third image (middle left), we have added *dimension-3* and isosurfaced at a value of 10 – this is *cell-(1,2,3).* Again a sense of *minima* near $(-1.0,1.0,1.0)$ and $(1.0,1.0,1.0)$

is gained. In the image middle right, we combine the view in dimensions $(1,2,3)$ with slice planes in dimensions $(1,2)$ – as already seen in step 2 of the exploration at top right – and in dimensions $(2,3)$. The $(2,3)$ slice passes through the larger of the two isosurface volumes, and does not show any other *minima*. This seems an interesting direction to pursue. Therefore we toggle *dimension-1* to give a slice through the space at the focus point of *dimension-1*, and this is shown lower left, confirming the view we had from the previous image. Finally selecting *dimension-4* allows us to enter the 3D space of dimensions 2, 3 and 4 – this is *cell-(2,3,4).* Bottom right shows an isosurface of value 10 in the *(2,3,4)-space*, indicating a tube structure containing the low values of the function.

We can proceed in this way, touring through the 4D space. We could switch to inspect around the starting point we used for the optimization codes, namely (-1.2, -1.0, -1.2, -1.0), by manipulating the focus point on the *n-dimensional Window* tool.

7.3.1.1 Exploring an optimization trajectory

In this section, we extend the example above to show how we can filter both multidimensional and multivariate data with the same tool. We are interested in seeing the trajectory of successive approximations to the minimum generated by a popular optimization technique, namely the Nelder and Mead simplex method [144]. We can regard the approximations as an ordered sequence of multivariate data items, to be displayed as a scatterplot. Therefore the dataset has four variates (the coordinates of a point in four dimensional space) and the number of observations is equal to the number of intermediate steps generated by the algorithm until it reaches the minimum. As before, we treat the visualization of the function itself as a multidimensional problem.

We create an IRIS Explorer dataflow pipeline as shown in Figure 7.6 – notice that the pipeline has two inputs: multivariate data representing the trajectory, and multidimensional data consisting of the function values on a grid.

Figure 7.7-(a) shows one stage of the investigation: two views (taken at distinct viewpoints) of the same 3D space comprising dimensions (2,3,4), corresponding to the final image in Figure 7.5. In addition to the visualization of the function – represented by an isosurface of value 50 – we have the outcome of the second visualization pipeline branch. This secondary pipeline inputs the simplex method – starting point $(-1.2,-1.0,-1.2,-1.0)$ – trajectory as 4-variate data, and applies a similar filter to generate a set of 3-variate data representing the trajectory – depicted by a sequence of spheres connected by line segments. Each sphere represents one step of the optimization algorithm, and its colour is associated with the order of the step in the sequence. Note that the last step is at the minimum, located at $(1,1,1,1)$.

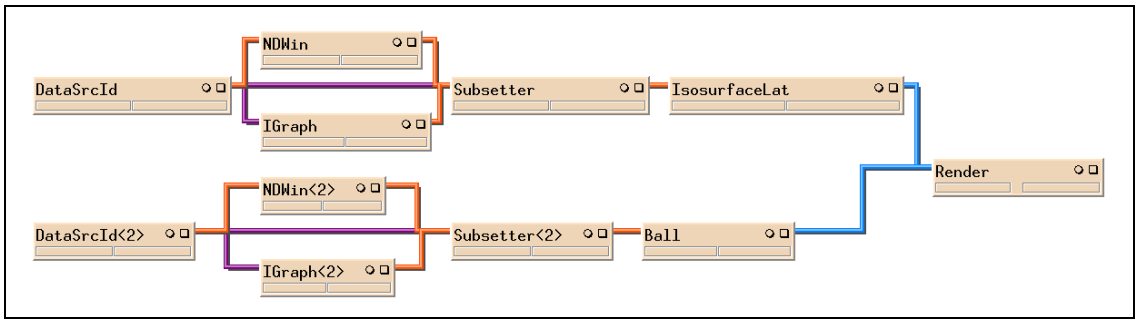
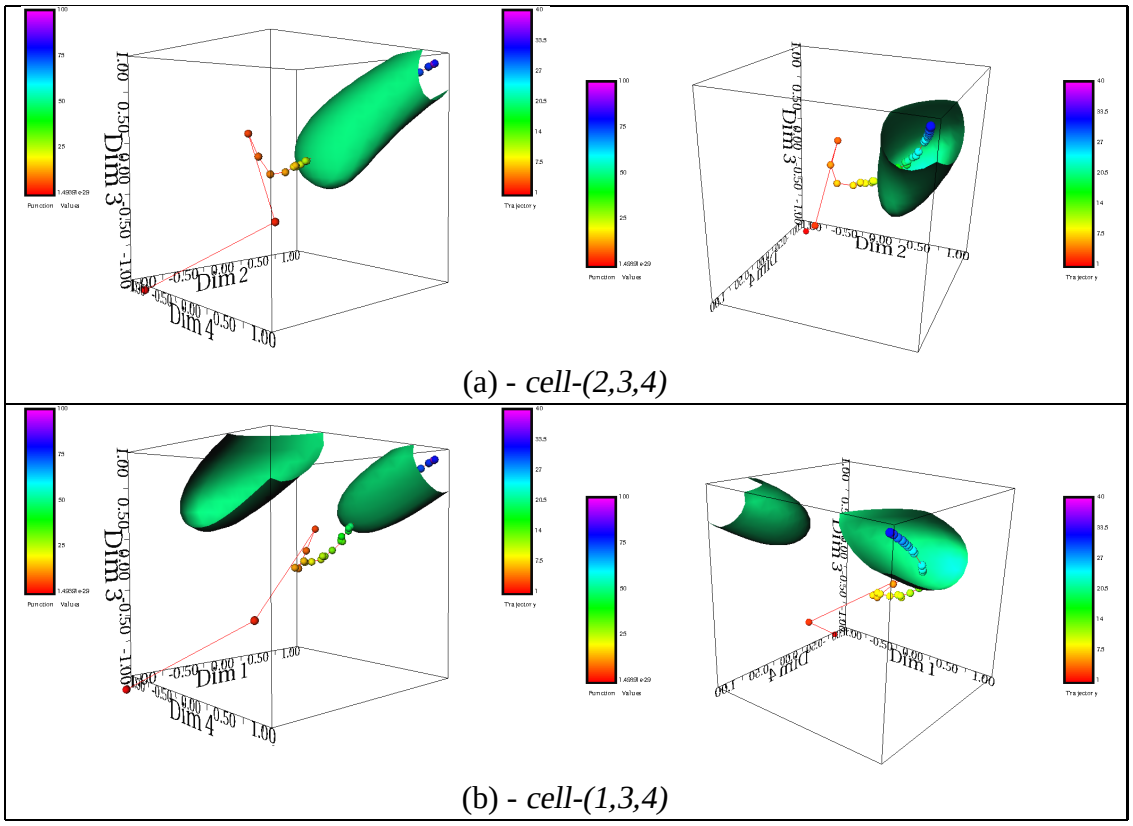


Figure 7.6: IRIS Explorer map showing two distinct pipelines, the top one for the multidimensional data (the 4D Rosenbrock function) and the bottom one for the multivariate data (the optimization trajectory). Both pipelines make use of the same type of modules for filtering the data. At the end the output of both pipelines are combined into a single visualization.



(a) - *cell-(2,3,4)*

(b) - *cell-(1,3,4)*

Figure 7.7: Combining the visualization of the 4D Rosenbrock function (multidimensional data) with the successive approximations to the minimum generated by the simplex method (multivariate data). Picture (a) shows two different views of *cell-(2,3,4)*, whereas picture (b) shows two different views of *cell-(1,3,4)*.

Note that in this case we have two instances of a filter, one to handle the visualization of the 4D Rosenbrock function and other to handle the visualization of the progressive trajectory of the minimum created by the simplex method. The trajectory, mapped as

3D scatterplots, and the geometry from the isosurface are merged into a single Render process. We can see the trajectory of the optimization algorithm as it enters within the isosurface of value 50, and progresses within that isosurface towards the minimum.

This visualization technique allows the algorithm developer to understand the way in which the algorithm converges to the solution, within the high-dimensional space. For example, switching to *cell-(1,3,4)*, and looking again at isosurface value 50, we get the images in Figure 7.7-(b). This figure shows the *cell-(1,3,4)* with two regions of low value (contrast with *cell-(2,3,4)* in Figure 7.7-(a) where there is only one region). Notice there are two areas of low function value (corresponding to arms of the '*n*-dimensional banana') but the algorithm is correctly following the downhill path to the region of lowest value.

However, during the examination of the Rosenbrock function and the application of the simplex optimization algorithm to find the minimum we noticed that for a certain starting point the algorithm was not able to find the global minimum. So we set out to investigate the following problem: *"Why does the simplex method fail to reach the global minimum of the four-dimensional Rosenbrock function when we provide the traditional starting point for optimization codes, namely* $(-1.2, 1.0, -1.2, 1.0)$*?"*

Figure 7.8-(a) shows the successive approximations to the minimum generated by the simplex method using the traditional starting point. The 'false' global minimum is located at $(-0.77, 0.61, 0.38, 0.14)$, and the function value at that location is 3.70142862. The global minimum, though, is located at $(1, 1, 1, 1)$ and the function value at that location is zero

This problem fits the *Scenario 2*, in which the user does not know *what* to look for (we do not know what is causing the algorithm to fail in finding the global minimum) but s/he knows *where* to look at (we know that we have to investigate the function behaviour along the trajectory corresponding to the successive approximations to the minimum).

We started the investigation by generating all possible 3D cells (four in total) and moving the coordinates of the focus point along the trajectory. As we move the focus point we observe the changes to the function values reflected in all cells, using isosurfaces at a fixed threshold. To help us to better understand what had happened during the execution of the algorithm we additionally plotted several points of the intermediate simplexes created during the interactions.

At a certain point along the path we noticed that the simplex got trapped inside a 4D *locus* in such a way that the expansion of the simplex was not able to escape that region. Consequently this made the algorithm assume that it had found the minimum. Figure 7.9 shows the visualization of the 4D Rosenbrock function at the 'false' global minimum, i.e. at $(-0.77, 0.61, 0.38, 0.14)$, with different isosurfaces. Note that the isosurface with value

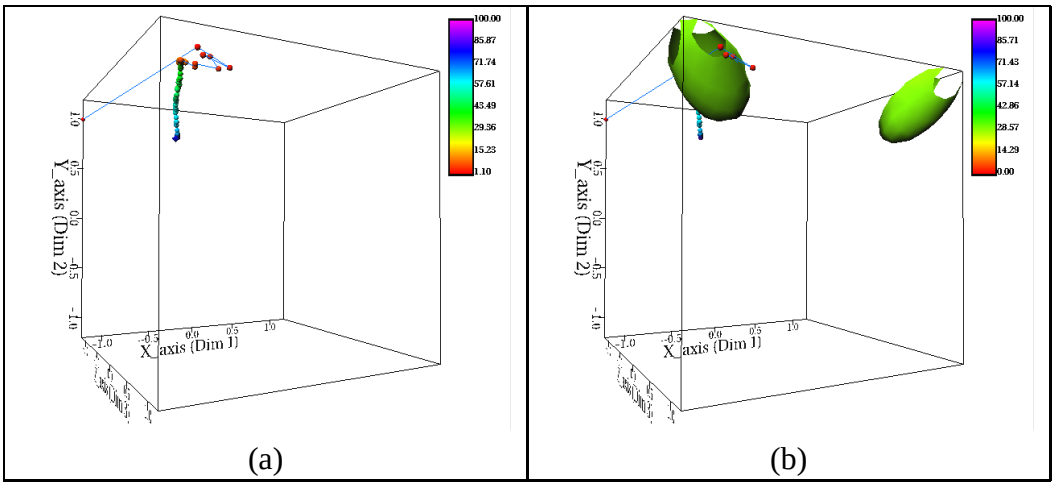


Figure 7.8: Visualization of the 4D Rosenbrock function combined with the successive approximations to the minimum generated by the simplex method using the traditional starting point $(-1.2, 1.0, -1.2, 1.0)$. Picture (a) shows that the trajectory does not reach the global minimum, and picture (b) shows an isosurface of value 30, which clearly shows the existence of another region that contains the global minimum, located at the far right corner of the *cell-(1,2,3)*.

2.0, shown in Figure 7.9-(b), is totally cut off from the real minimum at $(1, 1, 1, 1)$.

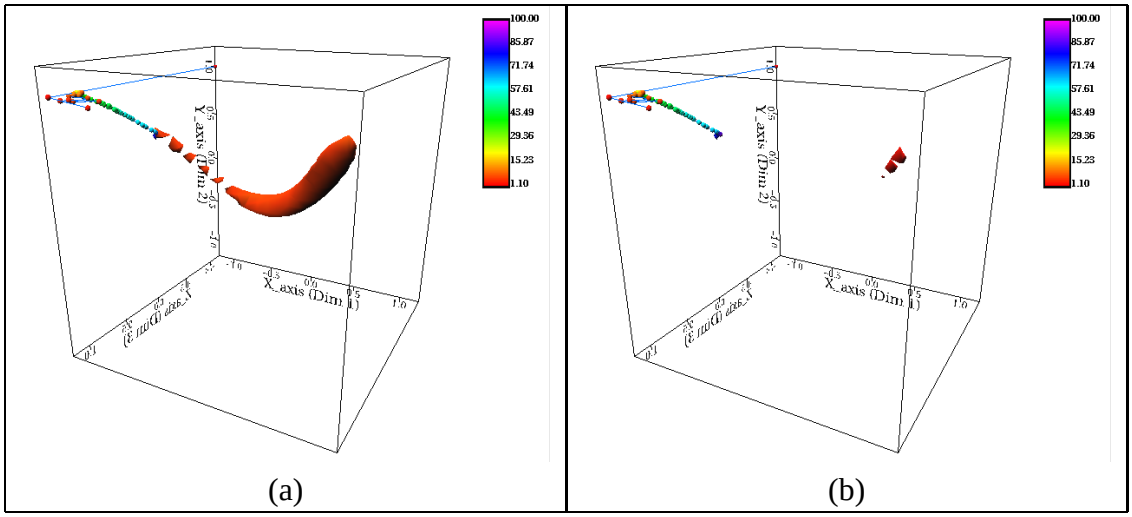


Figure 7.9: Further visualization of the 4D Rosenbrock function combined with the trajectory of simplex algorithm. Picture (a) shows *cell-(1,2,3)* depicting the trajectory combined with an isosurface of value 5, whereas picture (b) shows the same trajectory and an isosurface of value 2. Note in picture (b) how the 'false' global minimum is distant from the 'real' minimum located at $(1, 1, 1, 1)$.

We continued this investigation by creating a new dataset corresponding to a smaller 4D region around the 'false' global minimum. This visualization has confirmed that the

trajectory created by the algorithm had accessed a closed region in 4D similar to a 4D ellipsoid. Therefore we believe that this closed 4D region is responsible for the algorithm failure in finding the global minimum. Figure 7.10 shows the four 3D cells created around the 'false' global minimum visualized through an isosurface of value 5.0.

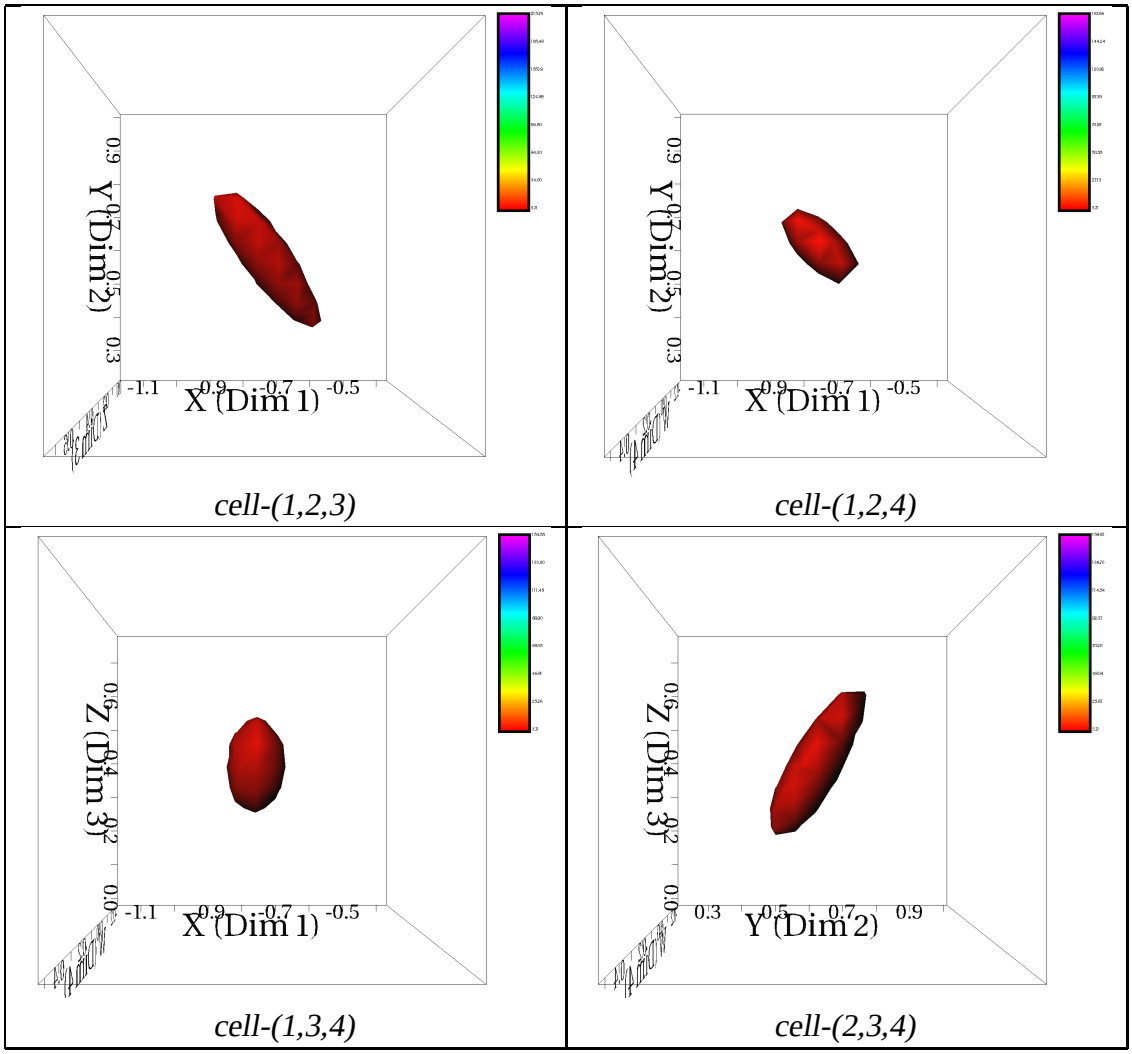


Figure 7.10: Visualization of the 4D Rosenbrock function around the 'false' global minimum found by the Simplex algorithm using the traditional starting point $(-1.2, 1.0, -1.2, 1.0)$. Note that in all four cells the 'false' global minimum (at the centre) is enclosed by an isosurface of value 5.0.

7.3.1.2 Remarks on case study #1

When one looks at a set of multiple views, new insights become possible. In Figure 7.11, we look again at Rosenbrock's function expressed by Equation 7.1, but this time in six dimensions (to illustrate also the way the approach scales to higher dimensions). In the

upper part of the Figure 7.11, we show the visualization in the *(4,5,6)-subspace*, the *n-dimensional Window* selection tool within the 6D space and the *Interaction Graph* selecting the dimensions 4,5 and 6.

However in the lower part we show all possible 3D subspaces – twenty in all. An interesting phenomenon is immediately visible – in all subspaces involving *dimension 1*, we see a second isosurface (at the other end of the 'banana'!) where the first coordinate is close to -1. In all other subspaces, the ten which do not involve *dimension 1*, there is just the single isosurface, enclosing the minimum point. By making dynamic changes to the *n-dimensional Window* and observing the result of this interaction throughout the cells we are able to explore this phenomenon in more detail.

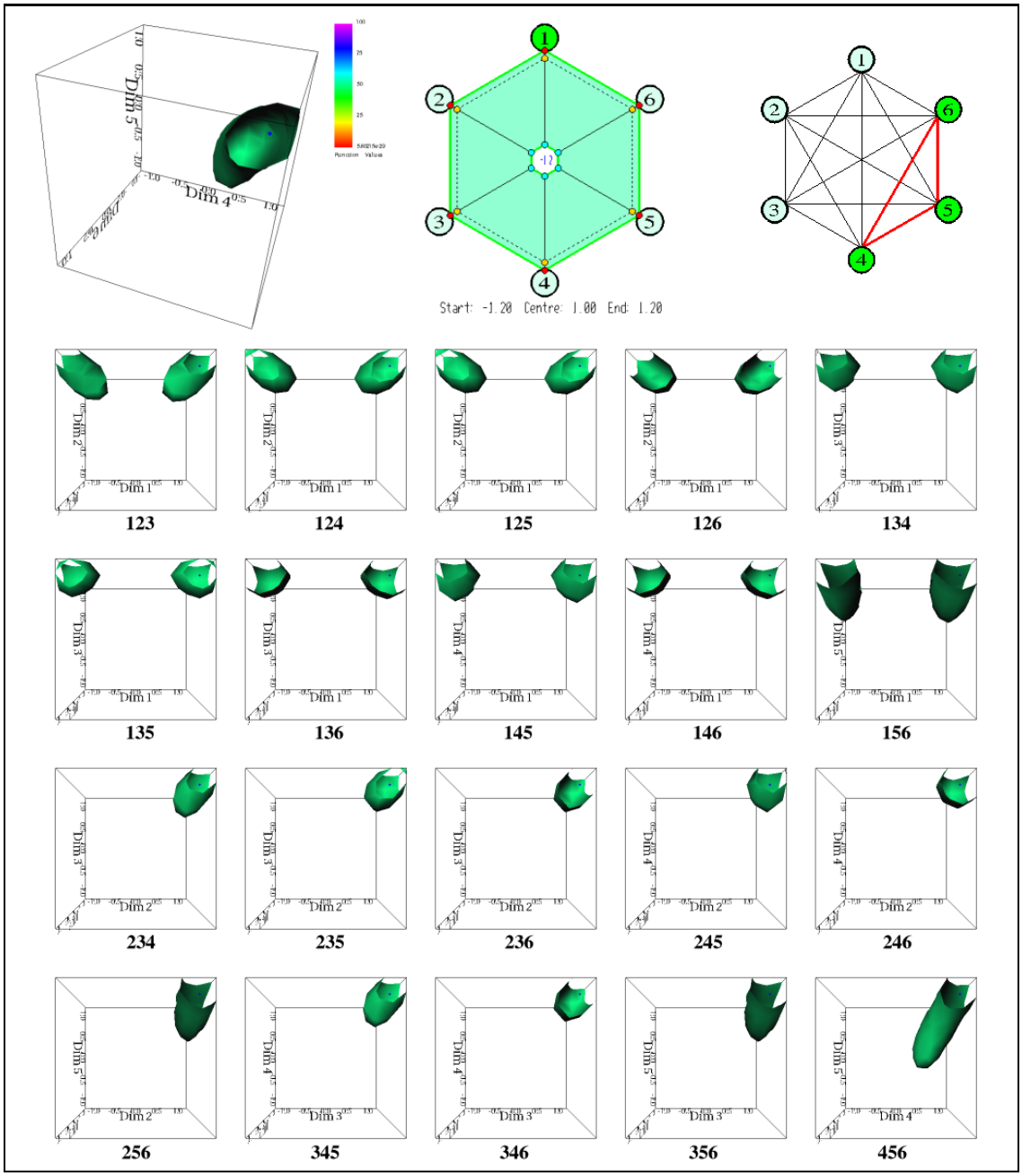


Figure 7.11: Showing a visualization of the 6D Rosenbrock function (multidimensional data) through all possible combinations of 3 dimensions, making a total of twenty distinct 3D subspaces. The three pictures on the top, starting from left are: visualization of a chosen subspaces, *cell-(4,5,6)*; the *n-dimensional Window* definition tool set for the 6-dimensional case; and the corresponding *Interaction Graph* having the dimensions 4, 5 and 6 selected. Just below them we have all the subspaces in a sub-sampled version to allow an overall view of the function. All the subspaces are obtained using the focus point $(1,1,1,1,1,1)$.

7.3.2 Case Study #2: Multivariate Astronomy Data

In this case study we are looking into astrophysical multivariate data organized as a data table of observations of several attributes taken from an astronomical catalogue – the SuperCOSMOS Science Archive (SSA) [89]. This is the database for a sky survey composed of four subsurveys – labelled *b*, *r1*, *r2* and *i* – denoting, respectively, a blue passband, two epochs of a red passband and a near-infrared passband. The database collects observations on sources in the sky which can be classified either as *stars* or *galaxies*.

We are using a particular dataset that is a subset of the original source database. This subset has 57 variates or attributes and a thousand observations (called sources) – 500 stars and 500 galaxies. The observations have already been classified either as stars or galaxies and the application goal is to help astronomers to perform the photometric calibration of the SSA data, i.e. to confirm whether the classification is accurate by looking at various expected correlations amongst variates and searching for unexpected ones.

This case study exemplifies a *Scenario 3*: the user knows what s/he is looking for (evidence to confirm whether the observations have been correctly classified as stars or galaxies) but does not have a clear idea where to find it (correlations between certain attributes are expected but others might be discovered that will help calibrate the data by providing the evidence needed to correct possible misclassification of sources).

This application problem seems particularly suited for *HyperCell* because the 57 variates can be nicely separated into groups of related attributes, the probable places to look for correlations. Therefore the filtering strategy is ideal to separate out these groups, allowing the astronomers to concentrate on groups of variates at a time. In this case study we present only three of these groups, which are representative for this application.

7.3.2.1 Describing some of the attributes

Here is a brief description of some attributes that are used in the examples to follow. The names of many of these attributes are identical except for the ending (*b,r1,r2,i*) – that is because this table collects together attributes for sources measured in the four subsurveys. The more meaningful attributes are the last 37 ones (21 to 57), 36 of which come in groups of four (for the four subsurveys).

The optimal way of performing the photometric calibration of the SuperCOSMOS data (i.e. to determine the most accurate measurements of the brightnesses of sources) depends, among other attributes, mainly on brightness measurements. Astronomers measure brightness using quantities called magnitudes, which are related via a negative logarithm to the physical brightness of the source.

So, one of the most important set of variates is composed of twelve attributes – the *classmags*, the *gcormags* and the *scormags* – these are the magnitudes of the source (in each of the four subsurveys) under the photometric calibration appropriate to astronomers' best guess at its classification (i.e. star or galaxy), and then calibrated as if it is a galaxy (*gcormag*) or a star (*scormag*). That best guess classification is *meanclass*, the next attribute, which is followed by the classification based on the data from each subsurvey – *classb*, *classr1*, *classr2*, and *classi*. Therefore the magnitude attributes and the associated classification are the variates 21 to 37.

Another attribute used to determine whether a source is a star or a galaxy is the ellipticity of the image – stars tend to be circular, while galaxies are often more elliptical. The next four attributes (*ellipb, ellipr1, ellipr2, ellipi*) are measures of the ellipticity of the source in four subsurveys (these are the variates 38 to 41).

The next set of four attributes (50 to 53)[2] – *prfstatb, prfstatr1, prfstatr2, prfstati* – indicates how point-like the image looks in each band. Prfstat (abbreviation for Profile Statistic) is a combination of several other attributes which together express how likely a source is to be point-like. So, there should be correlations between the ellipticity, classes and prfstats attributes, as all are conveying information about whether the source is star-like or not.

The last group we consider comprises four attributes (54 to 57), namely, *l*, *b*, *d*, and *ebmv*. The first two (*l*, *b*) are the location of the source in Galactic coordinates – i.e. in a reference frame in which the plane of the Milky Way is taken as the equator. The third attribute, *d*, is the distance (computed from *l* and *b*) of the source from the centre of the Milky Way (which is the origin of the Galactic coordinate system). Finally, *ebmv* is a measure of "extinction", which is basically how much attenuation due to dust you get along a particular line of sight in the sky.

Three groups of attributes are explored in the following subsections, as we provide some visualizations that have been used to help find correlations and calibrate the SSA dataset.

7.3.2.2 Group I: Location of the source in Galactic coordinates

We start by 'filtering' the last group (*l*, *b*, *d*, *ebmv*). The attribute *ebmv* should show spatial structure, since the distribution of dust in the sky is basically determined by the shape of our galaxy. Therefore the first selection of variates should involve the attributes *l*, *b*, and *ebmv*. This is shown in Figure 7.12 through a 3D scatterplot in which the observations

[2]The attributes from 42 to 49 have not been considered in this case study because they relate to encoded information that have not been used.

are mapped to spheres whose coordinates correspond to the three selected variates and the colour of spheres correspond to a fourth variate, namely *meanclass*. The *meanclass* indicates whether an observations has been classified as star or galaxy.

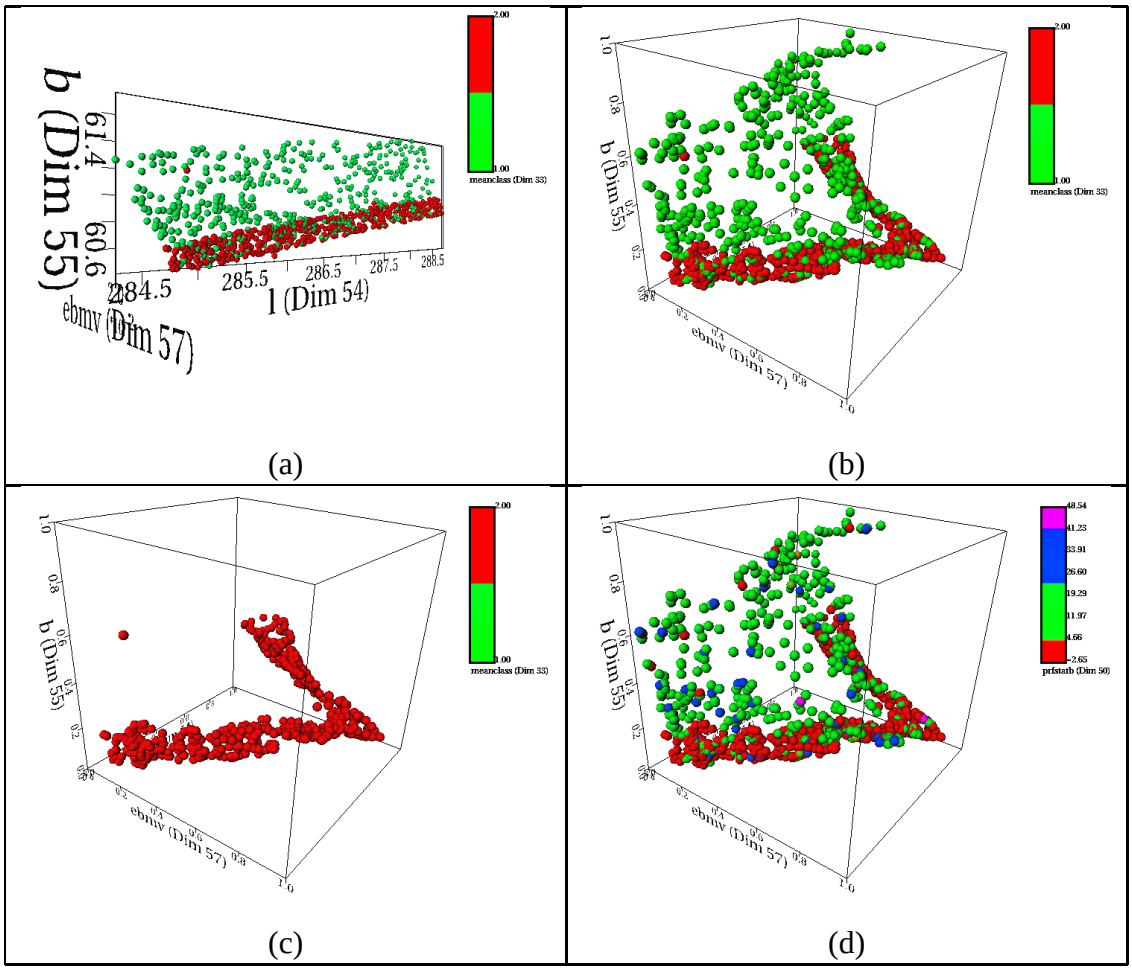


Figure 7.12: Visualizing *cell-(l, b, ebmv)* using a 3D scatterplot. Colour reflects the values of the *meanclass* attribute, indicating stars in red and galaxies in green. Note an 'outlier' star located at the top left corner of the cell, outside the 'band' of red spheres located at the bottom. Picture (a) uses the original values of the three attributes as geometric coordinates of the spheres, while in picture (b) the three variates' values have been mapped to the range $[0,1]$. Picture (c) presents the same cell; however the green class has been filtered out, making more evident the mentioned 'outlier'. Picture (d) also shows the same cell but coloured according to the *prfstatb* attribute, showing that in fact most of the candidates to be classified as stars are indeed located at the bottom, segmented in red.

Because the range of values for the attribute *ebmv* is smaller than the other two attribute ranges (i.e. *l* and *b*) the geometric dimension of the *visual space* associated with *ebmv* is compressed, making the 3D scatterplot look like a 2D scatterplot, as shown in Figure 7.12-(a). To overcome this situation the Subsetter module can be set to 'normalize' the coordinates of the *visual space* which means to map the range of values in each

variate onto the interval $[0, 1]$. The normalization produces a cell shaped like a cube that still preserves the relative values of a variate, as depicted in Figure 7.12-(b).

Notice in Figure 7.12-(b) an observation in red that does not belong to the red cluster at the bottom of the cell; instead this 'outlier' is located near the top left corner of the cell. This fact is made more evident in Figure 7.12-(c) after we have interacted with the *n-dimensional Window* and changed the range of the variate *meanclass* so that only the observations previously classified as stars are shown (the cluster in red). Therefore the visualization of Figure 7.12 has provided evidence for the fact that the particular marginal observation could have been misclassified as a star.

Finally Figure 7.12-(d) shows the same cell but this time coloured by *prfstatb*, which, as expected, shows most of the stars in red at the bottom of the cell. This confirms the fact that the *prfstat* attribute expresses how likely a source is to be point-like.

The investigation method used to look for correlations consists of selecting a cell, say *cell-(l,b,ebmv)*, and using the Subsetter to select different variates to colour the spheres. This operation is consistent with the one of the investigation strategies for *Scenario 3*, in which we choose a cell that is expected to produce a result and then we animate the focus point. However in this particular case we are in fact manually 'animating' the variate used to map the colour of the spheres, rather than the focus point.

We have also employed another suggested method for *Scenario 3* to investigate this cell (c.f. Chapter 5, Section 5.3.2) in which the *n-dimensional window* is manipulated to filter observations – this is, for instance, what we have done to create Figure 7.12-(c) showing only the star observations. Indeed the advantage of using *HyperCell* is more evident when dealing with the *n-dimensional Window* to filter out ranges of values for certain attributes and observing the corresponding results in the cells. For example, the astronomer may set the *prfstatb* value to a range that surely indicates potential stars and then check if the visualization provided corroborates that expectation.

7.3.2.3 Group II: Magnitude values of the sources

The most interesting correlations, astrophysically, concern the magnitudes, but these are typically studied via 2D scatterplots of differences in magnitude – e.g. *gcormagb-gcormagr1* vs *gcormagr1-gcormagi* in this case[3]. The reason for taking differences is that, since magnitudes are logarithmic measures of brightness, the difference between the magnitudes in two pass-bands is the ratio of the brightness in those two pass-bands, and, hence, a measure of the colour of the source. By working with colours, rather than mag-

[3] In fact the astronomers were quite interested in seeing how a 3D scatterplot involving further magnitude differences would look.

nitudes, one removes the overall scaling of the apparent brightness of an object – e.g. due to its distance from us.

One clear requirement for the development of visualization for this case in particular is the ability to pre-process data to plot differences of quantities against each other, not just the quantities themselves. Because *HyperCell* is developed within IRIS Explorer it is relatively easy to separate out 'channels' and feed them into modules that calculate differences – this pre-processing can be regarded as a *Data Analysis* task.

Therefore the next visualizations involve the colour-magnitude attributes. The first cell examined is the *cell-(classmag(b-r1), classmag(r1-i), classmagb)*, which is shown in Figure 7.13-(a). In that picture the *meanclass* variate has been mapped to the colour of the spheres in a 3D scatterplot. Note how the stars (red) and galaxies (green) lie in separate *loci*.

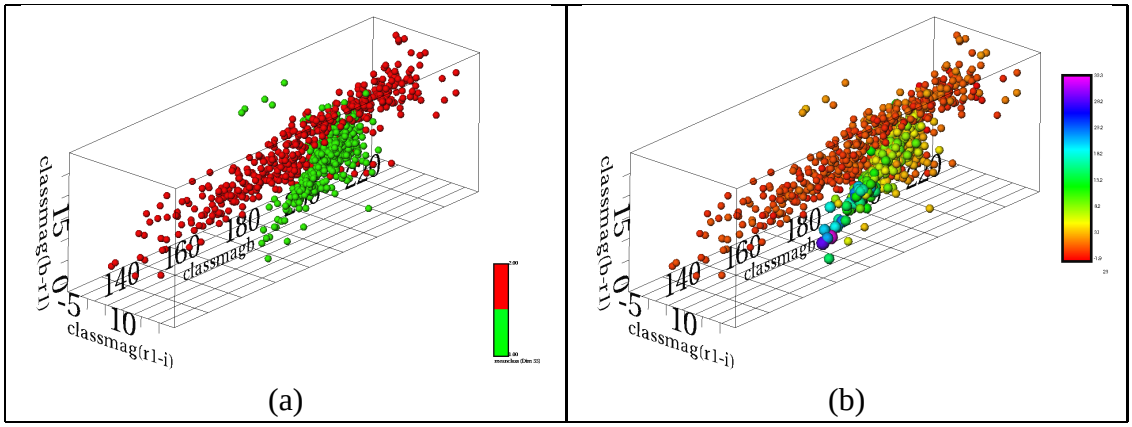


Figure 7.13: Visualizing *cell-(classmag(b-r1), classmag(r1-i), classmagb)*. In picture (a) the colour have been mapped to the *meanclass* variate, while in picture (b) the colour and size have both been mapped to the *prfstatri* attribute.

Considering the same cell we have found correlation involving both the ellipticities and the profile statistic attributes. The latter correlation is shown in Figure 7.13-(b) in which the spheres are coloured according to the *prfstatri* attribute. Note how the group of observations corresponding to the star cluster are represented roughly with the same reddish colour, indicating a low value of *prfstatri*. This is somehow expected because low values of *prfstatri* indicate a point-like source – thus prone to be classified as star. On the other hand high values of *prfstatri* are assigned to the cluster corresponding to galaxies.

The next pair of visualizations, depicted in Figure 7.14, represents *cell-(classmag(b-r1), gcormag(b-r1), scormag(b-r1))*. In Figure 7.14-(a) the colour represents the values of the *meanclass* variate. Note that again each class – galaxies and stars – lie on different planes that intersect along the main diagonal of the cell. Figure 7.14-(b) shows the same

cell from a different angle, but this time the colour attribute has been assigned to the variate *prfstatr1*. Note again the correlation between *prfstatr1* values and the magnitude differences in subsurveys *b* and *r1*.

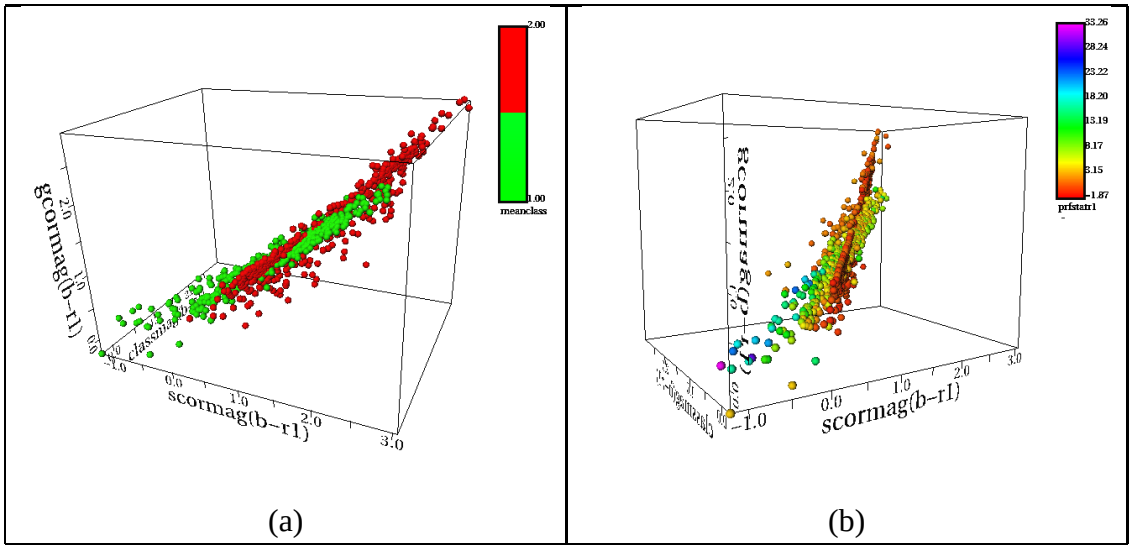


Figure 7.14: Visualizing the *cell-(classmag(b-r1), gcormag(b-r1), scormag(b-r1))*. In picture (a) the colour have been mapped to the *meanclass* variate, while in picture (b) shows the same cell from a different angle having both colour and size attributes of the spheres assigned to the *prfstatr1* variate.

7.3.2.4 Group III: Relating colour (magnitude) to shape attributes

Another interesting grouping suggested by the astronomers is to plot three of the profile statistic attributes and colour with *meanclass*. The stars should have much lower profile since that is the primary basis for separation between stars and galaxies.

This combination is presented in Figure 7.15-(a) that contains *cell-(prfstatb, prfstatr2, prfstati)*, having the colour mapped to the *meanclass* attribute. In that particular picture the separation between stars and galaxies is very clear through the two distinct clusters.

The same degree of separation, however, is not achieved by just looking at the cell made of ellipticities variates, as shown in Figure 7.15-(b). In that picture the two clusters intersect one another. It seems that the galaxies are more spread out in that cell, whereas the stars are concentrated at the origin (i.e. low values for the ellipticities attributes).

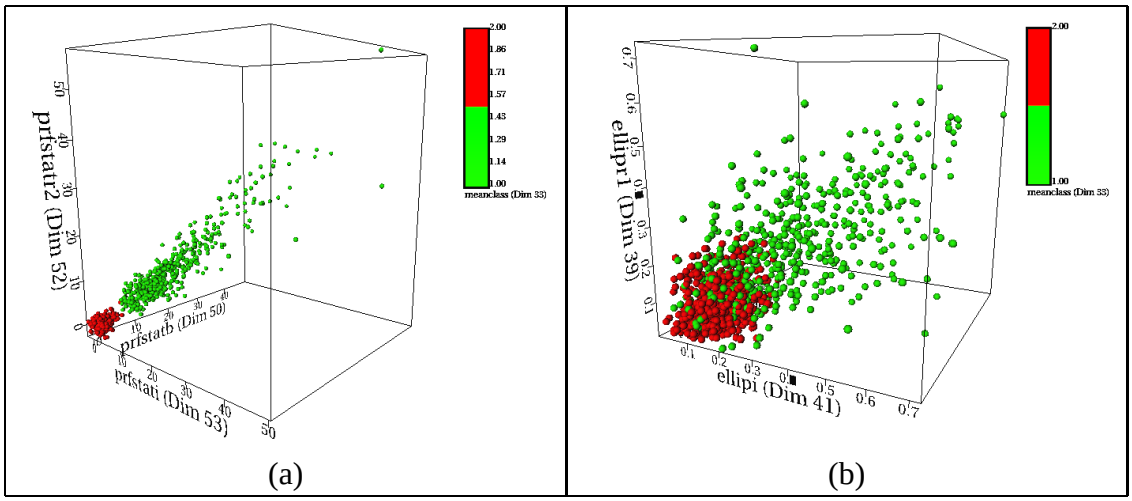


Figure 7.15: Picture (a) shows the visualization of *cell-(prfstatb, prfstatr2, prfstati)* using a 3D scatterplot. Colour mapped to the *meanclass* attribute, which confirms the *prfstat* attribute as a decisive attributes in classifying the sources into stars or galaxies. Picture (b) shows the *cell-(ellipb, ellipr1, ellipi)* coloured again by *meanclass*.

7.3.2.5 Remarks on case study #2

Interesting correlations have been found among the suggested subspaces, while others have been discovered during the exploration phase that are still being interpreted in astronomical terms (e.g. the two intersecting planes of Figure 7.14).

Several lessons have been learned from the application of *HyperCell* to this case study. The first one is that the power of *HyperCell* in dealing with three-dimensional cells can only be fully appreciated in interactive mode, which is not evident by looking at the static figures presented in this case study. These static figures were the end-result of the interactive manipulation of the cells.

Secondly, further information about the degree of correlation can be established if a more detailed investigation of the cells is performed. This would use interaction with the *n-dimensional Window* to restrict the ranges on variates to the values of interest. This approach certainly would allow the astronomers to verify any correlation depicted by the colour scale or even the spheres' diameter.

Thirdly, the mapping of a fifth variate to the diameter of the spheres used in the 3D scatterplots would, in theory, increase the amount of information covered. However we noticed that the use of these two sphere attributes – size and colour – to encode different variates is not very effective and, sometimes, it may even lead to confusion. We have found it more useful to use both colour and size of spheres to represent the same variate, following the recommendation for 'redundant mappings' [159].

Fourthly, we have confirmed that although this case study involved a considerable number of variates (57 in total), only certain combinations of variates (as indicated by the astronomers) really made sense for the investigator. This reinforces the strategy of letting the user control the creation of cells in contrast to generating and showing all possible combinations simultaneously. *HyperCell* has also scaled well to this high number of variates, especially the interface, as demonstrated early on in Figure 6.7 at the end of Chapter 6.

Additionally, the ability of using the concept of Dynamic Cell (c.f. Chapter 4, Section 4.4.1) to go from the traditional 2D scatterplots (which the astronomers are accustomed to understand) to the 3D scatterplots representation plus colour has improved the astronomer's exploratory power and yet managed to keep them comfortable because they can always return to the more familiar 2D representations.

Finally the best method for identifying correlations among the attributes was to create one or two cells involving the main variates of a group (for example, combining all four ellipticities attributes) and then dynamically changing the variate mapped to colour. The existence of simple linear correlations becomes immediately evident using this procedure.

7.3.3 Case Study #3: Multivariate Course Management Systems Data

Course Management Systems (CMS) are environments that support the development and the delivery of distance education courses over the Internet. The CMS are an important asset in providing support for many of the tasks required to run a distance learning course. However there still exist problems for the instructors to manage effectively all the aspects involved in a distance learning course.

A major problem the instructors face is to acquire a comprehensive understanding of the *social*, *cognitive*, and *behavioural* aspects of the remote students [91]. The main goal is to avoid common problems involving distance learning that have been repeatedly reported in the literature such as the *"lost in hyperspace"* syndrome (e.g. many alternative references on a topic are provided to the user, generating disorientation), the *"isolation effect"* (e.g. students want to communicate with their class mates or with tutors who could give advice on topics of interest), or the so-called *"I am neglected"* effect (e.g. learners need some sort of feedback on their progress) [91, 146].

The data generated by CMS can be used by instructors to help address those issues. Effective use of CMS as a supporting tool in distance learning would benefit from the design of suitable data visualizations, which in turn could provide the instructor with the evidence necessary to confidently answer questions like a) *how often did a student*

communicate with other students or the instructor?; b) *what is the performance of a given student on a specific topic?*; or, c) *how fast are the students progressing within the course material?* This would certainly help in the diagnosis of problems that might arise during the development of a distance course [133].

It is a common practice for the CMS to generate a complete log of all the students' activities, but this data has rarely been used to provide insight into these activities [133]. This is mainly due to the complexity of the data, which usually comprises several multivariate numeric and categorical data sources. Consequently the instructors normally discard this source of information because they find it difficult to extract any useful interpretation of the data just by looking at it in its regular tabular format – that is where visualization becomes an useful tool.

7.3.3.1 Data origin

Mazza and Dimitrova [134,135] have developed a visualization tool called CourseVis [133] that receives the data from a commercial CMS system called WebCT [207]. The CourseVis is a series of visualizations developed using the OpenDX system [152] as the front end visualization tool. CourseVis follows a list of recommendations compiled from a survey involving several instructors engaged in distance learning activities [132]. These recommendations are organized in *cognitive*, *social*, and *behavioural* aspects.

We have used the same set of recommendations as guidelines when using *HyperCell* to create the visualizations presented in this case study. We have also considered the feedback given by the instructors during the evaluation process of the CourseVis tool and our goal is to improve on the visualization results obtained with CourseVis.

Next we describe the nature of the three aspects – social, cognitive, and behavioural – and the recommendations for a successful visualization under each aspect. Then we present visualization examples following the recommendation for each of the three aspects, followed by an evaluation of the results obtained with *HyperCell.*

7.3.3.2 Describing the requirements for the CMS data visualization

The data used in our application come from an on-line course in Java Programming, which took place in 2002 at the Department of Informatics and Electronics of the University of Applied Sciences of Southern Switzerland[4].

[4]The data used in this case study was originally used by Mazza and Dimitrova, who have kindly provided us with the same data.

Social Aspects According to Mazza's survey [132] the most popular tools used to provide the communication facilities to an on-line course are *discussion forums*, *e-mail*, and *chat*. *Group work* has also been considered as a good device to stimulate social interaction and cooperation among the students of the course.

- **What instructors want:** To monitor the level of participation of students in the course and the level of interaction between students in a group.
- **What do they want to use the information for:** To gauge the level of participation of the students in the course and be able to use this information for the evaluation of both individual and course performances; and, to use this information to select parts of the interaction which would be analyzed qualitatively.
- **Recommendation:** The visualization representing the students' social activities should provide a quantitative visualization of the data from the communication facilities offered in the course. Therefore the recommendations are: a) the visualization should cover the participation in the discussion forum (Social Recommendation 1, or *SR*1); and, b) the visualization should depict the participation in group activities (Social Recommendation 2, or *SR*2).

Cognitive Aspects The cognitive aspect is considered the most important aspect. This aspect is directly related to the very objective of the course itself, which is to offer the opportunity for the students to learn the course's subject. They have also pointed out that the most popular assessment techniques are *assignment*, *quiz tests*, and *group work* in this order.

- **What instructors want:** To have information about the overall performance of the course; and, to measure the level of knowledge achieved by each individual on each domain concept of the course.
- **What do they want to use the information for:** To identify and remedy learners' common misconceptions in their courses.
- **Recommendation:** The visualization should: a) offer a clear and immediate external representation of students' performance (Cognitive Recommendation 1, *CR*1); b) help identify individuals having difficulties with a concept (Cognitive Recommendation 2, or *CR*2); and, c) be able to provide mechanisms for comparisons of individuals with the whole group (Cognitive Recommendation 3, or *CR*3).

Behavioural Aspects Behavioural indicators are usually useful in measuring factors like active learning, motivation, engagement, and even to assess the success or failure of a course. The behavioural information is usually data that reflects the students' access to the course material, their participation in the discussion forum and participation in group exercises.

- **What instructors want:** To have individual and collective information about access to the course material and the progress within the course schedule.
- **What do they want to use the information for:** To judge the mandatory participation in the course; to identify students progressing too slowly with the course schedule; to identify students with performance results significantly different from the mean of the group; and, to identify students who actively participate in discussion by posting messages and the opposite case, students who do not participate actively in discussions.
- **Recommendation:** The visualization should represent: a) the students' access to the content material (Behaviour Recommendation 1, or *BR*1); b) the submission and/or delivery of evaluation mechanisms by the students, such as quiz and assignments (Behaviour Recommendation 2, or *BR*2); and, c) the students' participation in discussion forum (Behaviour Recommendation 3, or *BR*3). Also the visualization should provide a graphic representation to work as an indicator of the students' progression within the course schedule (Behaviour Recommendation 4, or *BR*4).

Before we introduce the datasets used in each of the three aspects two observations are pertinent. Firstly, all categorical variates, such as names and dates, had to be mapped to a range of integer numbers because of the limitations of IRIS Explorer in dealing with categorical data. We recognize that this is not the recommended procedure according to traditional visualization guidelines, but this was the only option we had at that time to represent categorical data using *HyperCell.* Secondly, we have faced some difficulty in controlling the size of the labels used on the axes of cells – to change them we would have to select each individual character of a label and change the character's size manually by dragging its control boxes. Again this is a restriction imposed by the limitations of IRIS Explorer in dealing with non-numerical data.

7.3.3.3 Visualizing the social aspects

The dataset used for the visualization of the social aspects is multivariate, composed of the following variates:-

- *date*: Dates of the course, which started on 15/01/2002 and finished on 11/04/2002, making a total of 87 days. The dates have been mapped to the integer range $[1, 87]$.
- *student_id*: The students who participated in the course. There are a total of 17 students plus the instructor. They have been mapped to the integer range $[1, 18]$ to reflect an alphabetical order, and the instructor has been assigned to the last number of the range.
- *topic*: The range of topics on which a discussion thread can be initiated. There are a total of 16 topics, which have also been mapped to the integer range $[1, 16]$.
- *follow_up*: It is a scalar that reflects the length of a discussion thread, i.e., the number of replies a posted message has received.

The file reflects the data from a discussion board which allows the students to post messages to it. A message is composed of a sender, date, and a topic. The number of replies to a given message is called *follow-up* and it is expected that this value reflects the relevance of a given message. Figure 7.16 shows how the variates have been mapped onto the *n-dimensional Window* tool.

The visualization goals involving the behavioural aspect illustrate *Scenario 1* of the investigation scenarios (described in Chapter 5, Section 5.3). In the situation described by *Scenario 1* the user knows *what* to look for (the students' participation in discussion forum and group activities) and *where* to find it (this information can be obtained from the cell that involves the students, message topic, and date of posting).

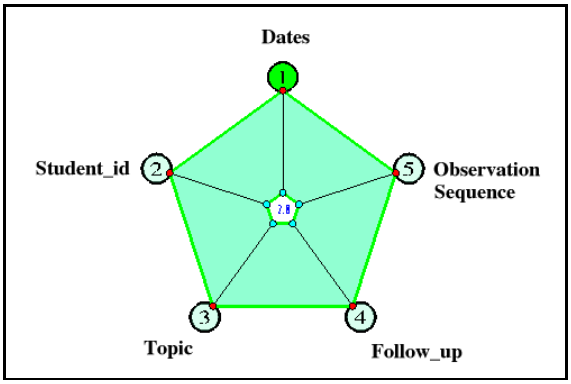


Figure 7.16: Mapping the variates' labels of the social dataset to the axes of the *n-dimensional Window* tool.

Figure 7.17 presents a 3D scatterplot visualization corresponding to *cell-(date, student_id, topics)*. The spheres' attributes, i.e. size and colour, are both mapped so as to

reflect the *follow_up* values. The goal of this visualization is to follow both *SR*1 and *SR*2 recommendations (c.f. Subsection 7.3.3.2), that suggest showing the participation of students in the discussion forum and group activities, respectively.

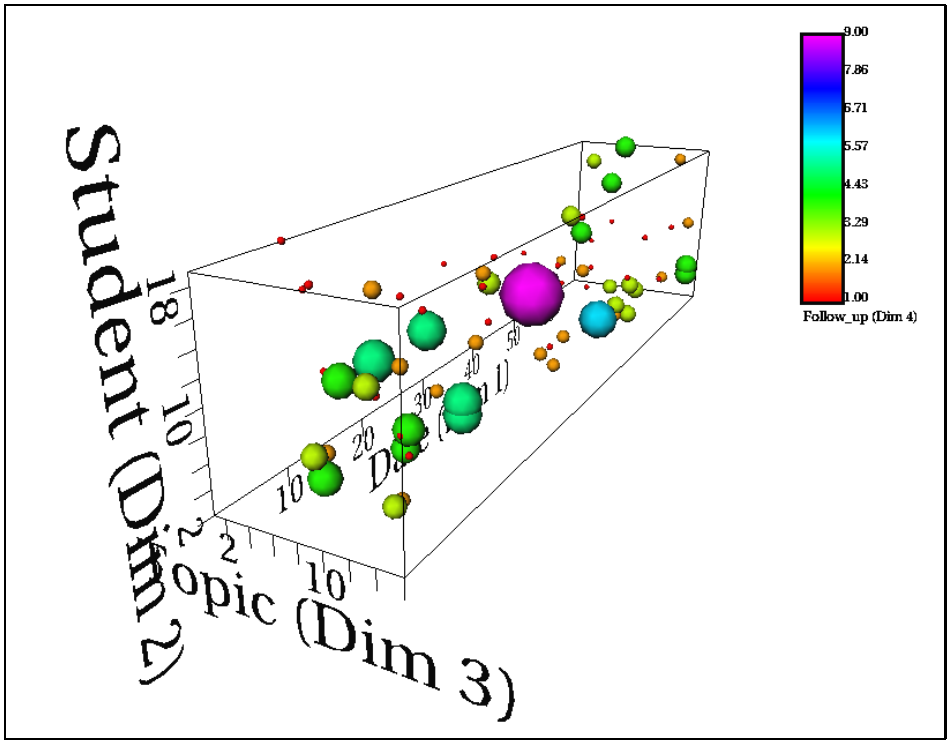


Figure 7.17: Visualization of *cell-(date, student_id, topics)* using a 3D scatterplot. The spheres represent the number of messages posted by the students throughout the course. The size and colour of spheres have been mapped to reflect the number of messages (variate *follow_up*).

A top view of *cell-(date, student_id, topics)* provides a better overview of the communication activities that happened during the course duration, as shown in Figure 7.18. This visualization informs that the course has had a fairly large number of messages posted which might be regarded as an indicator that a good degree of communication has been achieved. However this could only be confirmed upon a qualitative analysis of the messages' content.

In the next pair of visualizations the *n-dimensional Window* has been used to separate the communications related to the units of the course (Figure 7.19-(a)) from the communications related to the groups activity (Figure 7.19-(b)). The separation is accomplished by restricting the values of the variate *topics* on the *n-dimensional Window* diagram.

Notice in picture (a) of Figure 7.19 that there has been a fairly large amount of communication within each group. This visualization allows us to understand the message exchange between students of a work group. For example, by clicking on the spheres cor-

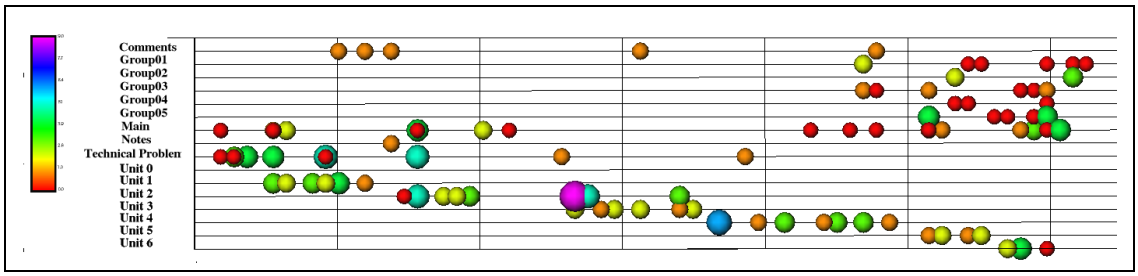


Figure 7.18: Top view of *cell-(date, student_id, topics)* shown in Figure 7.17. This view shows the variate *date* (horizontal axis) and variate *topics* (vertical axis). Labels on variate *topics* help the association of the spheres with their meaning.

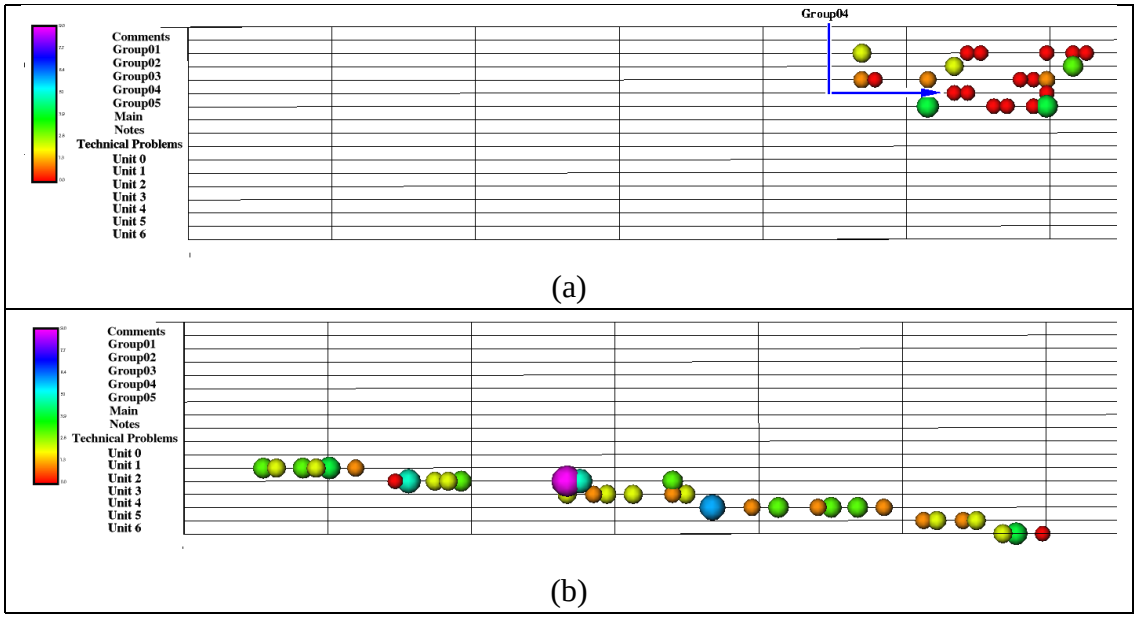


Figure 7.19: Further visualization of *cell-(date, student_id, topics)* (top view). In picture (a) only the messages within the groups have passed through the filter; while in picture (b) only the messages related to the units have passed through the filter.

responding to the Group04 (shown by a blue arrow in Figure 7.19-(a)) we could verify that these observations represent messages with no reply, i.e. the members of that group have not used the message board to communicate to each other. Finally picture (b) shows that the communication has been steadily developed throughout the course duration. In particular, no messages have been posted on the topic "Unit 0" – indicated by a line with no spheres on it; and the "Unit 2" has had the longest discussion thread, spread out through a longer period of time than the other units – indicated by the longest horizontal sequence of spheres that contains the largest purple sphere.

Looking at the same cell from a different angle, as shown in Figure 7.20, allows us to have an overview of the messages posted by individual students. This is important because the instructors need to be able to measure the level of participation of individual students and this is exactly what this new viewpoint can provide.

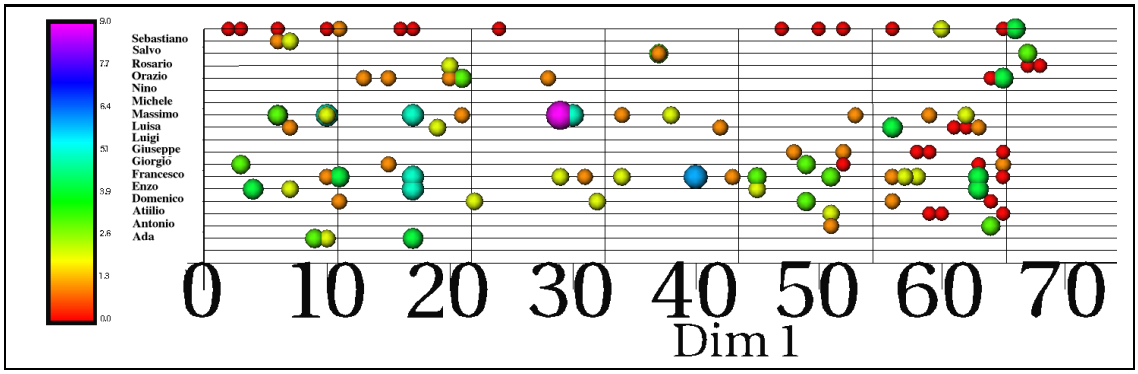


Figure 7.20: Side view of *cell-(date, student_id, topics)*. In this view the variate *date* is the horizontal axis and the *student_id* corresponds to the vertical axis.

The individual profile of the students can be generated again through the manipulation of the *n-dimensional Window* diagram so that a restriction is imposed on the *student_id* variate. This can be done either by creating multiple filters (each one isolating a specific student) or by using the NDNavigator to accumulate all the successive selection of students made with the help of the *n-dimensional Window* diagram and then using the NDNavigator to browse them.

A quick inspection of Figure 7.20 indicates that Francesco (*student_id* = 7) was the student who has posted the greatest number of messages, followed by Massimo (*student_id* = 11). Indeed a further investigation (i.e. isolating each student at a time) has revealed that Francesco posted 16 messages, whereas Luigi (*student_id* = 9), Michele (*student_id* = 12), and Nino (*student_id* = 13) posted no messages at all. Figure 7.21 shows a visualization that combined the results from two filters that have been set up to isolate the students Francesco and Massimo.

7.3.3.4 Visualizing the cognitive aspects

The dataset used for the visualization of the cognitive aspects has the following variates:-

- *student_id*: The same variate as in social aspect, representing the students who have participated in the course (17 students + instructor), mapped to the integer range $[1, 18]$ (the instructor is the number 18).

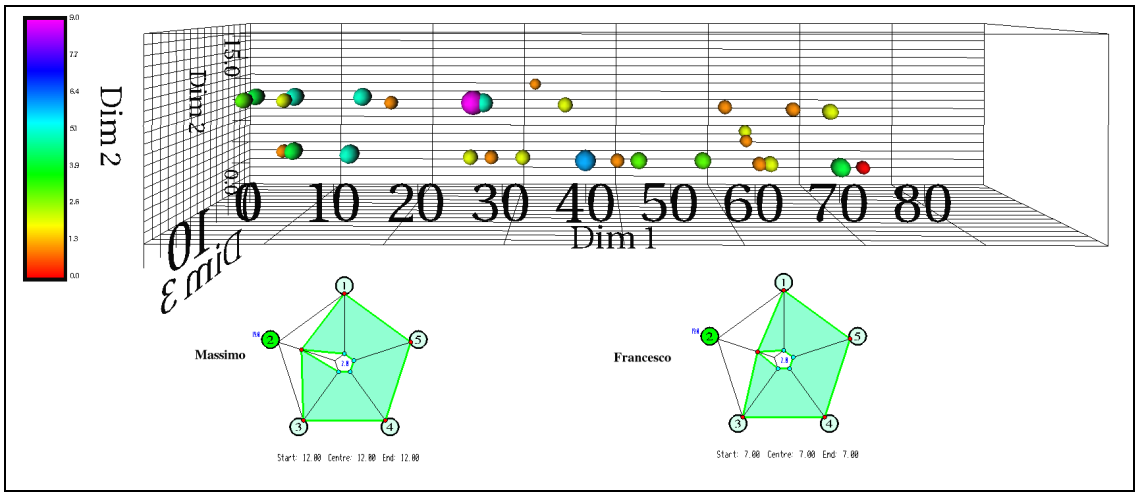


Figure 7.21: Side view of *cell-(date, student_id, topics)* showing the combination of two filters – one selecting the student Massimo and the other selecting the student Francesco.

- *concept*: Concepts covered by the course. They are basic units of the Java programming course making a total of 34 items, which have been mapped to a range of integers $[1, 34]$.

- *access*: Number of accesses a student has made to the course content on a given concept, represented as a non-negative integer that may vary from 0 to N, where N is the maximum number of accesses made. This parameter is a function of *student_id* and *concept*.

- *performance*: It is the mark a student has received when responding to a quiz on a specific concept, represented by a real scalar in the range $[0.0, +1.0]$. The performance is also a function of *student_id* and *concept*.

Basically the dataset used in this visualization is the combination of the quiz results and the number of accesses each student made to each concept of the course. Figure 7.22 shows how the variates have been mapped onto the *n-dimensional Window* tool.

Besides the original three recommendations regarding the implementation of visualizations of cognitive aspects (c.f. Subsection 7.3.3.2) we have also tried to answer the following question: *"Is there any relation between the number of accesses a given student has made to a specific topic and the student's performance on the same topic?"* In other words, the students who have accessed the course material a greater number of times might have an overall better performance than those students who have not accessed the course material so systematically. Again this characterizes the exploration *Scenario 1* introduced earlier: the instructor knows what to look for (in this case the individual and

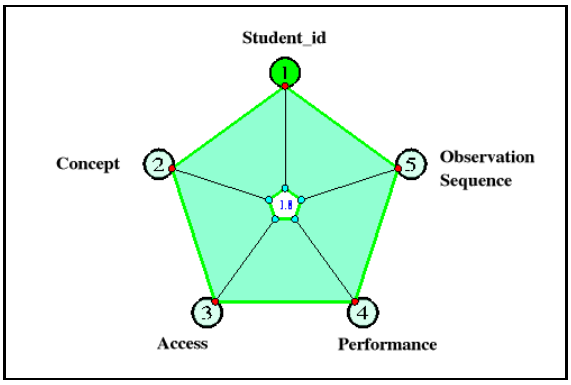


Figure 7.22: Mapping the original variates of the cognitive dataset to the axes of the *n-dimensional Window* diagram.

group performances) and has an idea where to look for it (the cells involving performance, accesses, students, and topics).

The first visualization attempts to follow the recommendation *CR*1, which suggests that the visualization should provide a representation of the students' performance. Figure 7.23 shows a visualization of *cell-(student_id, concept, performance)* through a 3D scatterplot. The colour and size of the spheres in the scatterplot reflect the values of the variate *performance*.

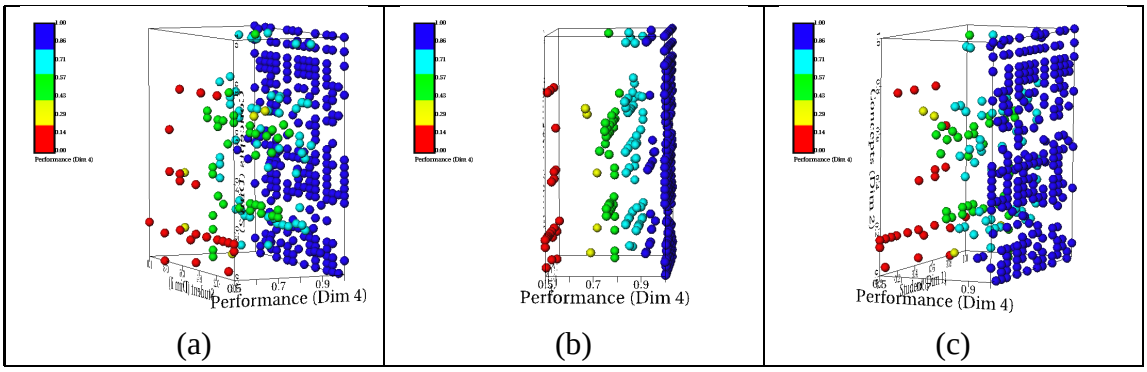


Figure 7.23: Visualization of *cell-(student_id, concept, performance)* through a 3D scatterplot. The colour and size of the spheres reflect the values of the variate *performance*. Picture (a) through (c) show the same cell rotating along the vertical axis (*concepts*) to make more evident the 3D aspect of the cell.

The first impression from the visualization in Figure 7.23 is that the group has had a good overall performance, judging by the majority of blue spheres. We have segmented the colour map used to map *performance* into five sequential ranges, represented by the colours red, yellow, green, cyan, and blue, respectively. Notice in Figure 7.23-(a) that

almost all students performed poorly in a certain topic (later identified as 'array'), which is shown by a horizontal sequence of red balls near the bottom of the cell.

The requirement *CR*2 emphasizes the importance of being able to identify students that might have been experiencing problems with a concept. The visualization in Figure 7.24-(a) attempts to address this requirement by showing *cell-(student_id, concept)* coloured by student performance. Individual performance can be examined by interacting with the *n-dimensional Window* tool to filter one student at a time.

Based on the visualization of Figure 7.24-(a) it is possible to identify those students (represented as columns) who have received low marks or have done only part of the quizzes; or the concepts with lowest marks. Here are some observations based on that visualization: 1) the student Enzo has done only two quizzes; 2) there are zero marks for Michele (in fact she abandoned the course); 3) Salvo has done just 8 quizzes; 4) Francesco has performed very well, followed by Attilio; and, 5) most of the students performed poorly on the 'array' subject (horizontal line in red near the bottom).

To highlight the concepts that have had the lowest marks the instructor could redefine the *performance* range on the *n-dimensional Window* tool, limiting it to, say, the range $[0.0, 0.4]$. The result is shown in the Figure 7.24-(b), in which the presence of a sphere indicates a low mark and the sphere's colour indicates the level of access: $[0,3)$ - red; $[3,7)$ - yellow; $[7,12)$ - green; $[12,18)$ - cyan; $[18,40]$ - blue. Also the visualization in Figure 7.24-(b) gives us evidence that there is no relation between a high number of accesses and high scores. Most of the students that appear in Figure 7.24-(b) (i.e. have low marks) have accessed the content several times (depicted by the spheres in yellow, green and blue). Finally, the visualization of Figure 7.24-(c) shows the number of accesses to concepts made by the students. Note that the student Massimo (*student_id* = 11) is the student who has accessed the course content the most.

7.3.3.5 Visualizing the behavioural aspects

The dataset corresponding to the behavioural aspects consists of the following variates:-

- *date*: Ordinal attribute that represents the dates of the course, which started on 15/01/2002 and finished on 11/04/2002, making a total of 87 days. The dates have been mapped to the integer range $[1,87]$,
- *student_id*: Nominal attribute representing the students who have participated in the course. Again the students have been mapped to the integer range $[1,18]$.
- *access_to_concept*: Nominal attribute that corresponds to the concepts covered by the course. They are basic units of the Java programming course making a total

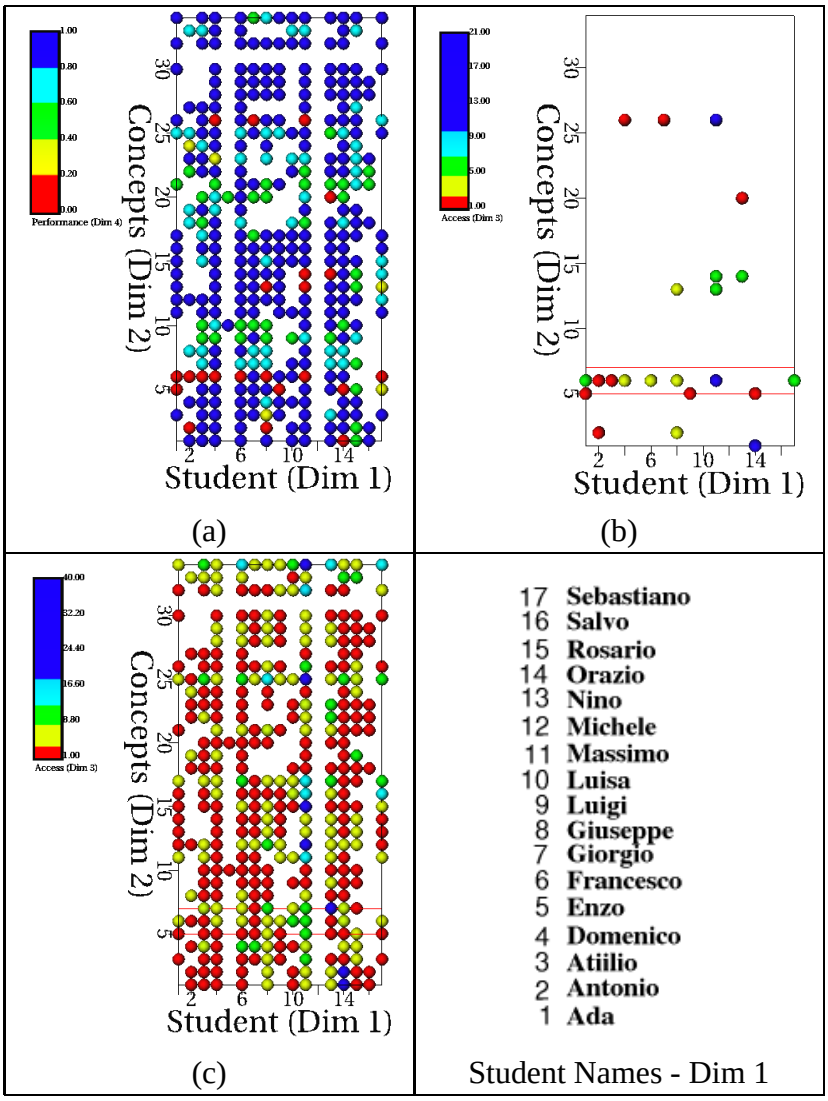


Figure 7.24: Visualization of the *cell-(student_id, concept)* via scatterplot. In picture (a) the colour of the spheres is assigned to the *performance*, segmented in five ranges (red, yellow, green cyan, blue). Picture (b) shows the same cell but coloured by the *access* variate, whereas in (c) the filtering has been set to let pass only those observations whose mark is below 0.4. Picture (d) shows the list of students associated with a list of integers.

of 34 items, which have been mapped to a range of integers $[1, 34]$. Every time a student accessed the content of a concept in the course material a entry was generated for that action. This information is particularly useful to know the sequence of concepts a particular student followed or to verify how often a student accessed basic concepts.

- *q_&_a_submission*: Nominal attribute that may assume two values Q (or 1) for quiz submitted; or A (or 2) for assignment submitted. This information may be useful

for the instructor in following the students' progress with assessed work.

- *message_type*: Nominal attribute that conveys information on the type of message posted by a student on a given date. It may assume three values, 1 for reading a posted message; 2 for posting a new message; and, 3 for replying to a message. Instructors may use this information to assess whether individual students are reading messages regularly, or participating pro-actively.

- *progress*: Integer number indicating the sequential number of a content page of the course. This sequential number represents the learning order of the course material. This information may be useful in identifying learning patterns such as those students who progressively and frequently read the course material on line compared with those students who would prefer to print off several pages at once and refrain from accessing the course material for a period. The value for this attribute lies within the range $[1, 104]$.

- *hits*: A positive integer value that represents the number of pages (or 'hits') a students accessed on a given date of the course. This information is useful in determining the access pattern of the group which, in turn, may help identify the periods of the course in which the students have accessed the course material more frequently (for instance because of a shortly due assessment). This attribute may assume values in the range $[0, N]$, where N is the largest number of hits a page has received during the course.

Basically the dataset used in this visualization is the combination of five files each of which having as the priority key the pair *date* and *student_id*. Therefore the last five attributes (*concept*, *quiz_&_assignment_submission*, *message_type*, *progress*, *hits*) are a function of the pair (*date*, *student_id*).

Note the following facts regarding the actions a student may take: (1) a student may access more than one concept on the same date; (2) a student may access more than one page on the same day (this referring to the *progress* attribute); (3) a student may generate a maximum of three messages of distinct types – article read, original post, or follow up – on the same date; and, (4) a student may perform both a quiz or an assignment on the same date. Figure 7.25 shows how the variates have been mapped onto the *n-dimensional Window* tool.

The first visualization is shown in Figure 7.26 which presents a 3D scatterplot of the *cell-(date, student_id, access_to_concept)*. The sphere's attributes, i.e. size and colour, are both mapped so as to reflect the *content* values. The goal of this visualization is to follow

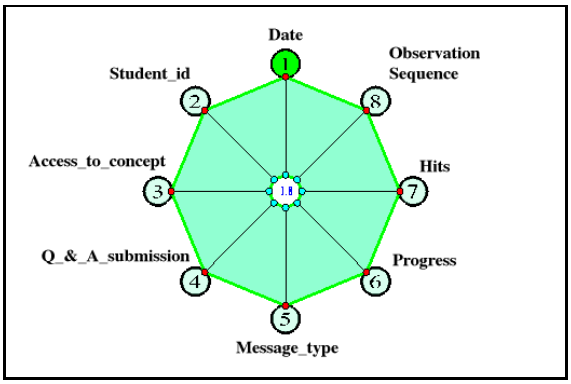


Figure 7.25: Mapping the original variates of the behavioural dataset to the axes of the *n-dimensional Window* tool.

*BR*1 recommendation, which suggests that a visualization should represent the students' access to the course material (c.f. Subsection 7.3.3.2).

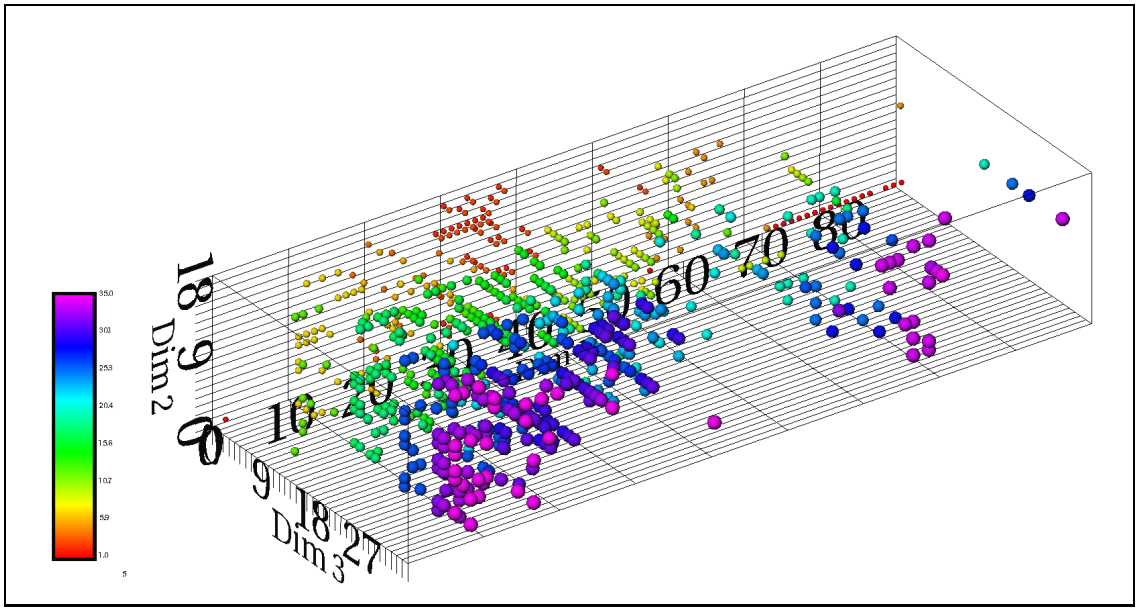


Figure 7.26: Basic visualization of *cell-(date, student_id, content)* using a 3D scatterplot. Both colour and size of the spheres have been both mapped to reflect the values of the variate *content*.

One way of obtaining an overview of the group and focusing on the sequence in which the students have accessed the concepts is to rotate the cell to get a top view, as shown in Figure 7.27. By doing so we concentrate on the behaviour of the whole group, focusing on *when* (instead of *who*) a concept was first and last accessed by the students.

In the visualization of Figure 7.27 the colour reflects the value of variate *date*, highlighting which concepts have been studied in parallel. There are roughly five groups of

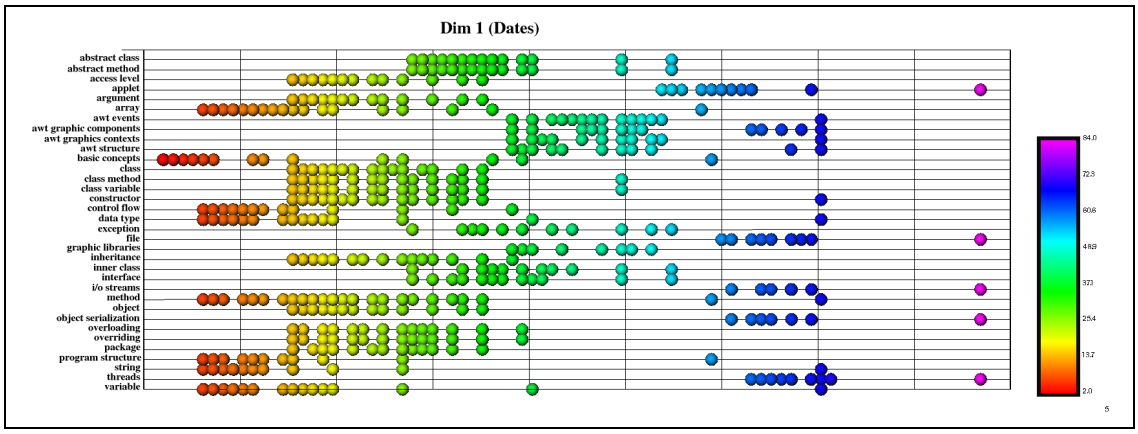


Figure 7.27: Visualization of the same cell as in Figure 7.26 but from a top view and having the colour of the spheres mapped to the value of the variate *date* (represented as the horizontal axis) to emphasize the groups of *concepts* (represented as the vertical axis) studied simultaneously.

concepts that have been studied in parallel. For example one group (represented by the reddish spheres) involves the following concepts: *array, basic concepts, control flow, data type, method, program structure, string*, and *variable*. Additionally, this visualization shows those concepts (the five purple spheres at the far right end of the image) that were reviewed by the students, namely *array*, *basic concepts*, *data type*, *method*, *program structure*, and *variable*.

The next visualization, depicted in Figure 7.28, still shows the same cell. Once more the *cell-(date, student_id, access_to_concept)* is seen from a different angle in which only the *student_id* and *access_to_concept* variates are visible. This view emphasizes the students (vertical axes) and how they have accessed the concepts (horizontal axis) throughout the course. The size of the spheres in the 3D scatterplot reflects the variate *date*, therefore a large sphere means that that concept has been accessed later than a smaller one. The advantage of this mapping is that we can verify, for instance, if all students are accessing the contents at the same time – this corresponds to a vertical line of spheres having approximately the same size. To illustrate this, consider the sixth column (yellow), which corresponds to the *awt events*. Now note that there is a late access to that concept made by Luisa, judging by the overlapping spheres with different sizes. In particular note the row corresponding to the student Massimo – he has reviewed some of the concepts, judging by the groups of overlapping spheres.

The next visualization attempts to comply with the *BR*2 recommendation (c.f. Subsection 7.3.3.2), which requires the visualization of the submission and/or delivery of evaluation mechanisms. For this particular distance course they have provided the stu-

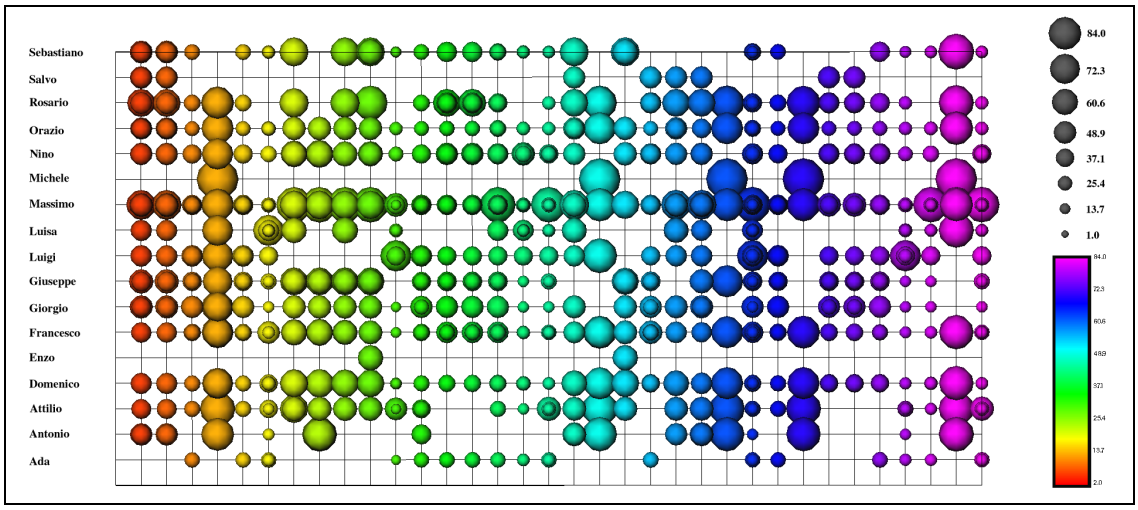


Figure 7.28: Visualizing *cell-(date, student_id, content)* from a angle that emphasizes both the *student_id* and *content* variates. The size of the spheres has been mapped to the *date* variate.

dents with two such mechanisms: assignment and quiz. Hence we have chosen the cell composed of the variates *date*, *students*, *Q_&_A_Submission*. Figure 7.29 shows this visualization, depicting the period in which the assignments (represent by the larger purple spheres) have been delivered: from 9/02/2002 to 20/02/2002 (corresponding to the period between 25 an 50 on the *Dates* axis); and a later submission on 5/03/2002 by the student called Rosario (the isolated purple sphere above mark 50 on the *Dates* axis). This may help the instructor to find out if the students have handed the assignment in before the due date.

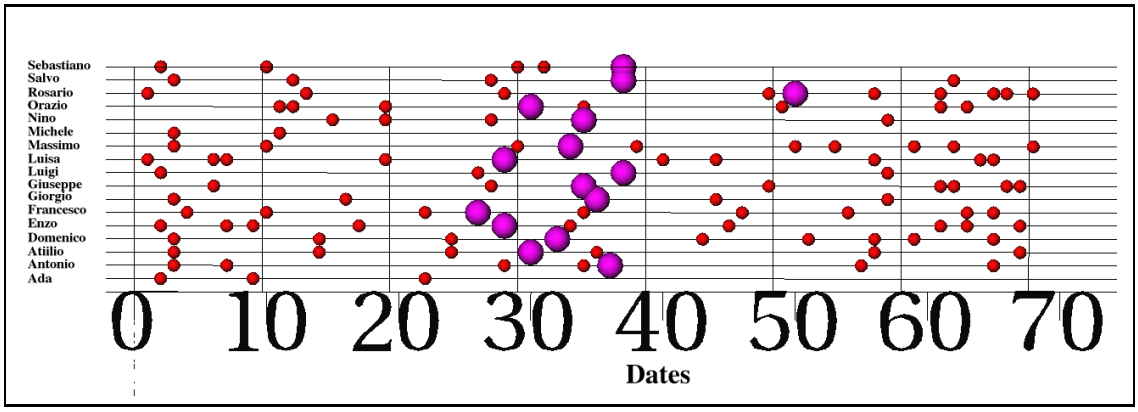


Figure 7.29: Visualizing *cell-(date, student_id, Q_&_A_Submission)* using a 3D scatterplot. The 3D cell has been rotated as to show a side view involving the variates *date* (horizontal axis) and *concept* (vertical axis). The spheres' attributes have been mapped to the values of the variate *Q_&_A_Submission*. This variate may assume two values, 1 for quiz submission (represented in red) and 2 for assignment submission (the larger purple spheres).

Note that the dates (as well as other attributes) corresponding to each sphere can be obtained by clicking on them – this makes the system print out a textual information with the data associated with the picked sphere.

The subsequent visualization follows recommendation *BR*3, which requires the presentation of the students' level of participation in communication activities. For that we have chosen *cell-(date, student_id, message_type)*, using the *message_type* variate to colour the spheres. The result is shown in Figure 7.30.

In Figure 7.30-(b) the cell has been positioned in such a way that the spheres are 'aligned', making possible to verify the frequency in which the students have generated a message and the type of message generated. The message types are: reading of an article (type 1, in red); posting a new message (type 2, in green), or; replying to an existing message (type 3, in blue). This view allows the instructor to promptly recognize those students who have been very active (such as Massimo, and Francesco) and those who have not (such as Ada, Michele, and Nino). Note also that it is possible to find those students who are very active in reading the message from others but not very participative in either posting or replying to messages, namely Luigi and Sebastiano.

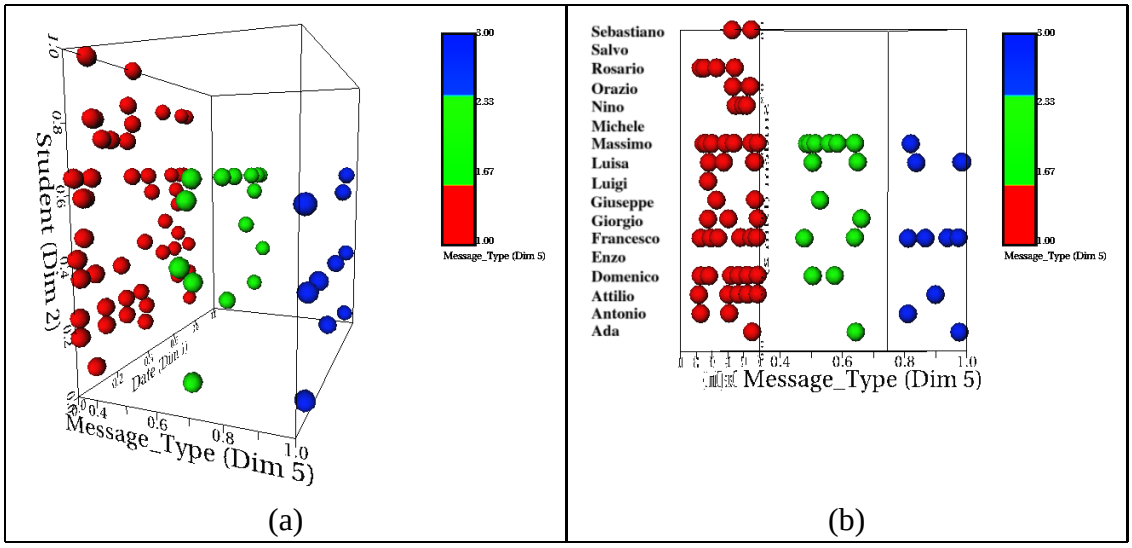


Figure 7.30: Visualization of *cell-(date, student_id, message_type)* using a 3D scatterplot. In picture (b) the cell has been moved so as to achieved a 'compression' effect because of the orthogonal projection used to represent the 3D cell. Red spheres represent *article reading*, green spheres means *posting new message*, and blue spheres indicates *replying to an existing message* type of event.

Finally the last visualizations are related to the recommendation *BR*4: to visualize the progress of the students within the course schedule. Figure 7.31 depicts *cell-(date, student_id, progress)*, having the colour of the spheres mapped to the values of the variate

progress and their size to the variate *hits*. The overall result shows that the course has followed an expected behaviour, with the students accessing the content pages in the expected sequence (note the 'staircase' pattern). The bigger spheres indicate those periods in which the number of 'hits' to the website were more intense, such as at the beginning of the course and at the end.

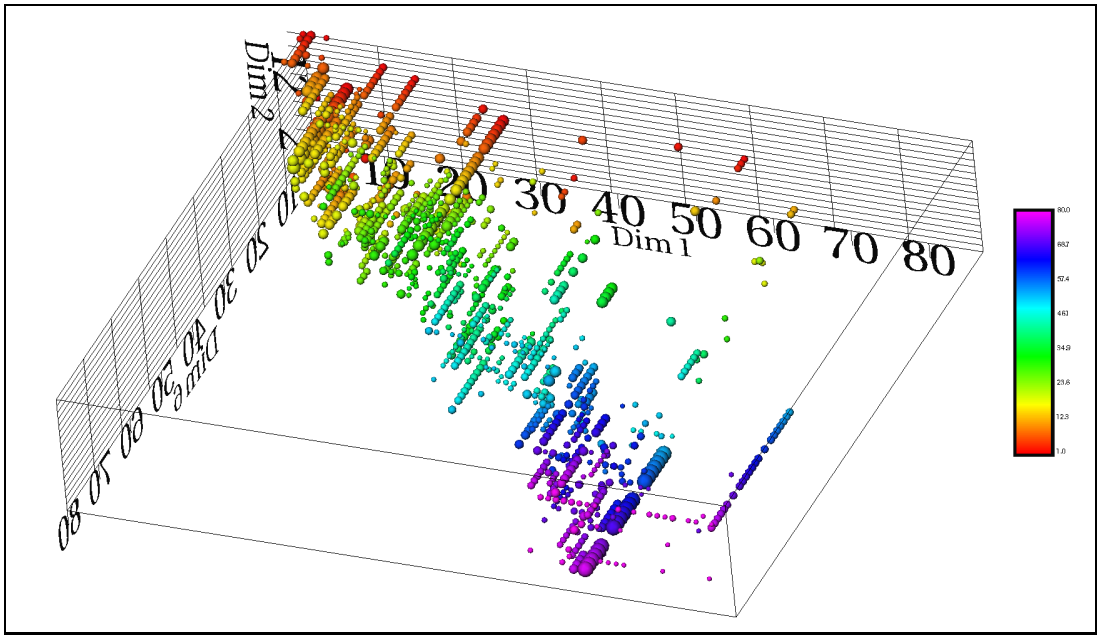


Figure 7.31: Visualization of *cell-(date, student_id, progress)* using a 3D scatterplot. The colour of the spheres has been mapped to the value of the variate *progress* and the size of the spheres has been mapped to the values of the variate *hits*.

7.3.3.6 Remarks on case study #3

Throughout Section 7.3.3, devoted to the third case study, we have presented some basic visualizations, aiming to provide an overview of the *HyperCell* potential. Most of the visualizations produced in this case study fit into the *Scenario 1*, in which the user knows what she or he is looking for (the recommendations found in each of the three aspects), and where to find them (the cells usually involve the student identification, the date and a third or forth variates). This makes it easier to use *HyperCell* to generate visualizations, because the interesting cells are known beforehand.

At the end of the experimentation with the CMS data we have been able to comply with every recommendation identified by Mazza and Dimitrova (c.f. Section 7.3.3.2), with the added bonus of providing the various visualizations through a common environment. Furthermore the visual representations used to convey information are consistent in the

sense that they all employ a similar representation (3D scatterplots), thus reducing the cognitive load of interpreting distinct visual representations.

Additionally, Mazza and Dimitrova in [135] have listed some issues that a group of instructors have identified as important for the visualization of CMS data, after interacting with the CourseVis system. We believe that *HyperCell* can provide valuable solutions for these issues, listed below:-

- **Social aspects.** Instructors requested more representations to monitor how often a student has posted messages, or who is attending discussion vicariously[5].

  HyperCell solution: The visualization of Figure 7.30 shows how frequently students have posted messages. By manipulating the *n-dimensional Window* tool to restrict the range on the the *student_id* variate it is possible to visualize one student at a time. The vicarious students have been identified by the visualization in Figure 7.30-(b).

- **Cognitive aspects.** Instructors requested a link between the cognitive visualizations and the social and behavioural aspects. This would help in finding, for example, reasons for a poor performance in a given concept.

  HyperCell solution: The link between the three aspects is possible through the use of multiple filters, one for each dataset corresponding to the three aspects. To compare a specific student regarding, say cognitive and social aspects, it is necessary to manipulate the corresponding *n-dimensional Window* to 'filter' the same *student_id* value. Thus the visualizations resulting from this coordinated manipulation of the filters would show the cognitive and social aspects of the selected student.

- **Behavioural aspects.** The visualization provided by the CourseVis regarding the behavioural aspects was considered to provide more information than needed, thus increasing the cognitive load in understanding the graphical representation. The recommendation for this problem is to allow the instructors to select which data should be presented.

  HyperCell solution: This problem is avoided when applying *HyperCell* because usually only 3 variables are selected to compose an individual cell. Of course, extra variables can be added if the spheres' attributes are considered. For the particular case of the behavioural aspects the instructor would have to create a cell that involved only the pair (*student_id-date*) plus the variate for which a comparisons is

[5] Vicarious interaction happens when a student actively observes and processes the interaction between members of the course without taking an active role in that interaction [190].

necessary. By creating several of these cells it would be possible to compare distinct attributes from the behavioural aspects – thus no extra information is added unless it has been requested.

7.4 Final Remarks

Regarding the case studies, we have chosen these datasets for two reasons. Firstly, they are representative of the exploration scenarios (see Table 7.1 below) presented early on in Chapter 5 (Section 5.3). Secondly, they typify classes of applications in the following sense:-

- Case study #1 can be applied to the visualization of any scalar function of many variables, to the sensitivity investigation of optimization problems represented by this type of function, and to the analysis of the execution of optimization algorithms. Furthermore this is a representative example of multidimensional data in scientific visualization domain (continuous model of data).

- Case study #2 represents a class of problems that involves numeric multivariate data, whose objective is to identify clusters and outliers, and correlations. Therefore this could be applied, for example, to the Iris dataset type of problem, which characterizes a multivariate dataset of scientific origin. Also this example has been useful in exploring the scalability of *HyperCell* due to the large number of variates.

- Case study #3 is a typical representation of abstract data (not physically based and discrete model of data), involving a mixture of numerical and categorical variables.

Table 7.1 summarizes the three case studies described in this chapter, emphasizing the type of data and the exploration scenarios as described in Chapter 5.

Case Study	*Data Type*	*Data Category*	*Area*	*Exploration Scenarios*
Rosenbrock Function	multidimensional	numerical	SciVis	Scenario 4
Simplex Optimization Trajectory	multidimensional and multivariate	numerical	SciVis	Scenario 2
Astronomy Data	multivariate	numerical	SciVis	Scenario 3
CMS data	multivariate	numerical and categorical	InfoVis	Scenario 1

Table 7.1: Summary of the three case studies.

Some of the lessons learned from the use of *HyperCell* are:-

- The manipulation of the *n-dimensional Window* is a powerful device in exploring the data, especially when coupled with the NDNavigator which enables the user to go back to previously stored interesting NDWin configurations. For example, when dealing with the CMS data it is useful for the instructors to be able to visualize the data of individual students. In this case the instructor can use the *n-dimensional Window* tool to 'filter' individual students and their data, while storing the corresponding filter configuration plus annotation in the NDNavigator module. To recover that particular visualization the instructor would only need to click on the corresponding representation of the filter (on the NDNavigator's user interface) associated with a student and send that configuration back to the *n-dimensional Window*, so that only that student appears in the visualization.

- The mapping of categorical data to integer numbers works fine but needs to be improved so that the user can recognize the original labels associated with each category. The main problem is a result of the chosen implementation platform – IRIS Explorer. This MVE system does not have the same functionality for categorical data as for numerical data, mainly because the system is originally designed to deal with scientific (numerical) data.

  Therefore representing categorical information just as numbers hinders the visualization potential. One possible solution is to build a module that receives the labels and exhibits a window showing a table with the association between numbers and the labels of the categorical variates.

- The use of the 'fruit machine' lay-out strategy to organize the cells was useful in the case study #1 (Rosenbrock function), especially when moving the focus point along the simplex trajectory.

- With relation to the rule of *attention management* we noticed that when various cells are created in the course of the *n*-space exploration they tend to obstruct the view of one another. Therefore it would be interesting to promote a stronger link between the manipulation of the *n-dimensional Window* via the NDWin tool and the cells associated with it. This could be done, for instance, by making all cells of a workspace (i.e. linked to the same *n-dimensional Window*) to stand out every time the *n-dimensional Window* is modified via NDWin's interface.

- For the *linking and brushing* method to be fully appreciated, *HyperCell* should allow a direct manipulation on the *visual space* to create the brushing object which, in turn should be propagated to all other linked cells.

The overall impression of using *HyperCell*, obtained from the informal feedback given by the experts involved in these case studies, is very encouraging. *HyperCell* has been successful in providing insight into the high-dimensional data involved in these applications when compared to early attempts to perform the same tasks. In the 4D Rosenbrock function case we have compared our tool to the situation in which the investigator tries to visualize the same function and trajectory using only the tools available in the IRIS Explorer system; in the astronomy case we have compared the results obtained with *HyperCell* with the results obtained by employing simple 2D static scatterplots (the usual procedure adopted by the astronomers); and, with the CMS case study we have been able to compare most of our results with the visualization provided by the CourseVis system.

The challenge remains in establishing a 'quantification' for the level of insight, which is certainly a difficult task. Perhaps the implementation of controlled experiments focusing on specific tasks related to the exploration of, and the navigation within the *n*-space could offer some starting point for measuring usability.

7.5 Summary

In this chapter we have appraised the *HyperCell* visualization technique by applying the following two-step strategy: to investigate how well *HyperCell*'s design complies with a set of guidelines in comparison to an early version; and, to systematically apply *HyperCell* to real applications, working with the experts involved[6] (so called *usability inspection*) in order to compare the results obtained with *HyperCell* with the results from the usual tools used to understand their data.

The current *HyperCell* design has benefit greatly from the selected guidelines, especially those defined for multiple coordinated views. The testing phase was also favourable. We have applied the technique to three case studies involving several types of data and from different backgrounds (see Table 7.1 for a summary). These case studies covered all four investigation scenarios described early on in Chapter 5, thus allowing us to verify the value, in terms of fostering insight, of the exploratory methods suggested in that section.

The impact of the evaluation procedure was twofold: 1) it has helped us to identify strong points of *HyperCell* such as uniform treatment of multidimensional and multi-

[6]The experts are mentioned in the Acknowledgements of this dissertation.

variate datasets, the power of the filtering strategy in reducing the complexity of high-dimensional data, and the scalability of the interface (especially when dealing with the astronomy data); and, 2) we have obtained a positive feedback from the experts involved in the three case studies, supporting the assumption that the proposed strategy of exploring high-dimensional space is a valuable one.

Chapter 8

Conclusions

THIS DISSERTATION HAS aimed to describe a visualization technique for multivariate and multidimensional data that follows a uniform framework based on the data-flow paradigm. The proposed framework can also be used as a basis to better understand high-dimensional visualization techniques.

Ultimately a visualization method for high-dimensional data comes down to the task of finding a suitable mapping to 'translate' the high-dimensional *data space* – regardless of being either multivariate or multidimensional – into a low-dimensional geometric *visual space*. Hopefully, such mapping will be easier to represent on the screen than the original data, and will also bring insight. Indeed the major distinction between the various existing visualization techniques is in the way they create such a mapping.

To better understand this process we have revisited the influential reference model for visualization, proposed by Haber and McNabb over a decade ago, from the perspective of multidimensional and multivariate visualization. The goal was twofold: to identify the common elements present in the process of visualizing high-dimensional data; and, to seek ways of refining and/or adapting the core elements of the model in such a way that minimizes the distinction between multivariate (normally associated with Information Visualization) and multidimensional (normally associated with Scientific Visualization). Therefore the ultimate goal of this research was to find a foundation model that incorporates both types of data into a common visualization process, thus bringing InfoVis and SciVis rather closer together, at both conceptual and practical levels.

8.1 General Summary

In Chapter 1 we have introduced our basic terminology, aimed at reducing misinterpretation of terms such as variables, dimensions, variates, multidimensional, and multivariate. In this chapter we have also described the research problem, our motivation, and the goals of this research.

Throughout Chapters 2 and 3 we have focused on the research problem on a higher level. We have firstly reviewed various approaches designed to classify and, consequently, understand high-dimensional visualization methods in both InfoVis and SciVis. From this examination we have been motivated to develop our own classification mechanism called the Three Stage Visualization (TSV) ontology. The basis of the TSV classification is the three common stages that we have identified from the several methods reviewed: *data analysis*, *data picturing*, and *data interaction*. We have found the TSV ontology a valuable means to understand the visualization methods in terms of these three stages. This also has the side effect of allowing us to compare methods and evaluate their strengths.

The next step was to pursue a common framework based on the three visualization stages and on the Haber–McNabb dataflow model, so successful in describing scientific visualization techniques. In that model – usually described as a sequence of three processes, Data Enhancement, Mapping and Rendering – the visualization mapping and rendering processes have been the main focus of research in the field and are now well understood. Rather less attention, however, has been paid to the data enhancement process. The original intent was that it should be an interpolation process, for example generating a regular grid of data from a given set of scattered data. In reality it has often been regarded as a step that selects data of interest from a larger initial set. In our suggested model the data enhancement step has been refined into a pair of processes: *data analysis* and *data picturing* – both elements of the TSV ontology. We have also introduced an extra component, influenced by InfoVis visualization models: *data interaction*. We have found that the model would benefit from the data interaction component, designed to account for the activity in which the user engages when fine-tuning the end result of the visualization process.

In Chapter 4 we have made our case for using the *filtering* strategy as our first choice for the *data picturing* step. Some of the advantages are: 1) variable filtering removes some of the dimensions from the display, thus avoiding cluttering that may occur if all variables are displayed simultaneously; 2) variable filtering is more intuitive to the user than, say, approaches that rearrange the variables and, consequently, lose the original variable values; 3) variable filtering is flexible – the user interactively selects or unselects variables to

be filtered; 4) filtering tends to be more effective because it generates a number of simple, easy to understand displays, each focused clearly on a particular aspect of the underlying data; 5) by using filtered subsets with low-dimensionality (i.e. up to four variables) it is possible to rely primarily on spatial mapping – one of the most valuable forms of visual encoding – to visually present the data; and, 6) most of the techniques based on other strategies promote the distortion of the *n*-dimensional relationship between data points in order to map the data to the display, whereas filtering-based techniques explicitly attempt to preserve these relationships. The filter step itself is separated into two further processes, one defining an *n*-dimensional window within the space, the other making a selection of variables for display. A key aspect of the *filtering* strategy is the uniform treatment of both multidimensional and multivariate data.

A novel visualization method, called *HyperCell*, has been developed following the suggested framework and adopting the *filtering* strategy. The *rationale* behind *HyperCell* is to provide the user with tools that support the creation of several low-dimensional subspaces, called 'cells', corresponding to the filtered data. Chapter 4 has focused on the process of setting up the filter, describing the several tools involved, namely IGraph – used to select the variables to be filtered; NDWin – responsible for defining a *n*-dimensional window within the high-dimensional space; Subsetter – designed to extract the filtered subspace from the original data source.

Chapter 5, in turn, describes how these various subspaces can be pieced together to form a cognitive model of the high-dimensional data. This involves 1) the *coordination of the subspace* – task assigned to the Workspace Manager module; and, 2) the *definition of several strategies to explore the n-space* to achieve the visualization goals.

1. *Coordination of the subspaces*: The paradigm described in this paper is one of sequential exploration of the high-dimensional datasets, through successive low-dimensional subspaces, with a smooth transition between these subspaces.

   This paradigm has been extended to allow multiple filters or concurrent views, where we retain views of where we have visited in our previous explorations. A nice feature of this approach is that dynamic changes to the *n*-dimensional window are propagated to all elements of this 'array', and so all visualizations generated from the array are dynamically changed. The experience is of walking through the *n*-dimensional space (by moving the focus point for example) and seeing the effect in all the subspaces previously created – corresponding to looking around in different directions in the *n*-dimensional world. Multiple filter processes allow the behaviour in multiple *n*-dimensional windows to be studied simultaneously, and has

the advantage of allowing us to keep track of all visited locations in *n*-dimensional space.

2. *Investigation scenarios for the n-space:* The complexity of exploring high-dimensional spaces has been summarized in four possible investigation scenarios. These scenarios have been defined based on the prior knowledge of the user regarding *what* to look for in *n*-space and *where* to find the interesting features or behaviour.

   Based on these scenarios several exploratory methods have been suggested. One of the important elements related to exploration is the navigation of the *n*-space – a task handled by the NDNavigator module.

HyperCell has been implemented in terms of the visualization system, IRIS Explorer, as a set of new modules. By integrating into an existing environment (rather than building our own standalone tool), we immediately gain access to the rich functionality of that system. The reasons for choosing a MVE system together with implementation details have been discussed in Chapter 6.

Finally, Chapter 7 describes the evaluation procedure we have followed in order to appraise the suggested visualization technique. We firstly compared the system design to a group of guidelines related to multiple coordinated views and visualization issues. We explained how the present design of *HyperCell* had resulted from improvements to an earlier version. These improvements were driven by the need to increase conformance to the guidelines. However the design evaluation has shown us that the *linking and brushing* tool can still be improved. In particular, the system lacks a more emphatic perceptual mechanism to draw the user's attention on the cells that are being manipulated via NDWin (especially when a large number of cells have been created).

Then we proceed to demonstrate the visualization system by applying it to three case studies covering multivariate and multidimensional data from InfoVis and SciVis backgrounds. Through the use of *HyperCell* we have gained an understanding in all three applications in comparison with the visualization tool normally used in each case, as well as being able to cover all four investigation scenarios proposed in Chapter 5. The overall result is, thus, very encouraging, both regarding the design evaluation and the usability inspection done in the case studies.

8.2 Summary of Contribution

The present work makes several contributions to the principles and practice of the visualization field:-

- The definition of a new classification scheme, called the *Three Stage Visualization* (TSV) ontology, for high-dimensional visualization techniques. This proposal is based on a clear definition of categories that represent three distinct phases of the visualization process: *data analysis* – in which the data undergo a transformation process (often a mathematical one), aimed either to reduce the data dimensionality, or to prepare data for mapping; *data picturing* – in which a visual mapping strategy tries to further reduce the dimensionality, thereby making it accessible to users; and *data interaction* – which describes the various mechanisms used to access the components of the visualization network, either to control the parameters of the intermediate data stages or to manipulate the rendering to enhance the final outcome.

  The TSV ontology has proved to be a valuable tool to help comparing the various visualization techniques in terms of these three visualization stages, as well as providing the basis for a framework for the development of high-dimensional visualization methods.

- The outlining of a general framework that describes a uniform approach for both high-dimensional abstract and scientific data. The idea is to abstract the differences between the visualization of multivariate and multidimensional data, and concentrate on the common elements – thus bringing (at least part of) information visualization and scientific visualization rather closer together. Several attempts have already been made towards this goal as described in [52, 62, 101, 119].

  The level of integration offered by the suggested framework contributes to the visualization field by providing the basis for extending the use of modular visualization environments in the information visualization domain. Additionally, the framework encourages the re-use of algorithms originally designed for scientific data into abstract applications (for example, it is possible to use an isosurface method to isolate multivariate observations in a scatterplot representation, thereby working as a 'brushing' device by assigning one of the variates to be the value of an associated function).

  Finally, the framework supports the understanding of visualization techniques in terms of the three stages of the framework. For instance, the framework proposed in Chapter 3 has enabled us to describe another new visualization technique we have been working on: *3D Parallel Coordinates.* Albeit at an early stage, this new method has been implemented as a prototype and has produced encouraging results when applied to the multivariate Iris dataset, as shown in Figure 8.1. This technique

plots the observations of multivariate data in a similar way to the original parallel coordinates – observations are mapped to polylines connecting points across parallel axes representing the variates – the major difference is that the polylines now have a third spatial dimension, being, therefore, plotted along an axis orthogonal to the plane formed by the parallel axes (i.e. coming "out" of the screen). The idea is to use the sequence of observations to define the order in which the polylines are plotted along this new third dimension ('Z' axis). Different ordering of observations produce different sequences of 3D parallel coordinates and the aim here is to rely on the human perceptual ability to recognize patterns in the 3D polylines as we rotate and move them within the *visual space*, while experimenting with different ordering.

Other proposals to extend the functionality of parallel coordinates have been put forward such as manipulating the order of the variables [40], handling hierarchical structures [73, 179], extruding parallel coordinates [208], and using curves instead of polylines to improve perception [81].

- A novel visualization technique, called *HyperCell*, which has been designed based on the proposed framework. This technique is our attempt in proving the value of the suggested framework and has been successfully employed in three case studies involving multivariate and multidimensional data, from both abstract and scientific origins.

  Advantages of this new technique are: uniform treatment of both multivariate and multidimensional data; scalability – each additional variable increases the number of possible subspace but the control over their creation remains with the user; adaptable visual representation – the user is free to employ any simple visualization techniques for the cells; and, re-use of visualization algorithms originally designed for the scientific visualization domain on information visualization methods and vice-versa, hence reducing the gap between these two domains.

8.3 Future Work

This work has touched on the very difficult task of looking for visualization methods for high-dimensional data that can increase the level of understanding of the *n*-space in which the data, whether abstract or physically grounded, is defined.

At the end of this work we have proposed a solution for the problem that we have found of value, backed up by the evidence from the design evaluation, and the usability

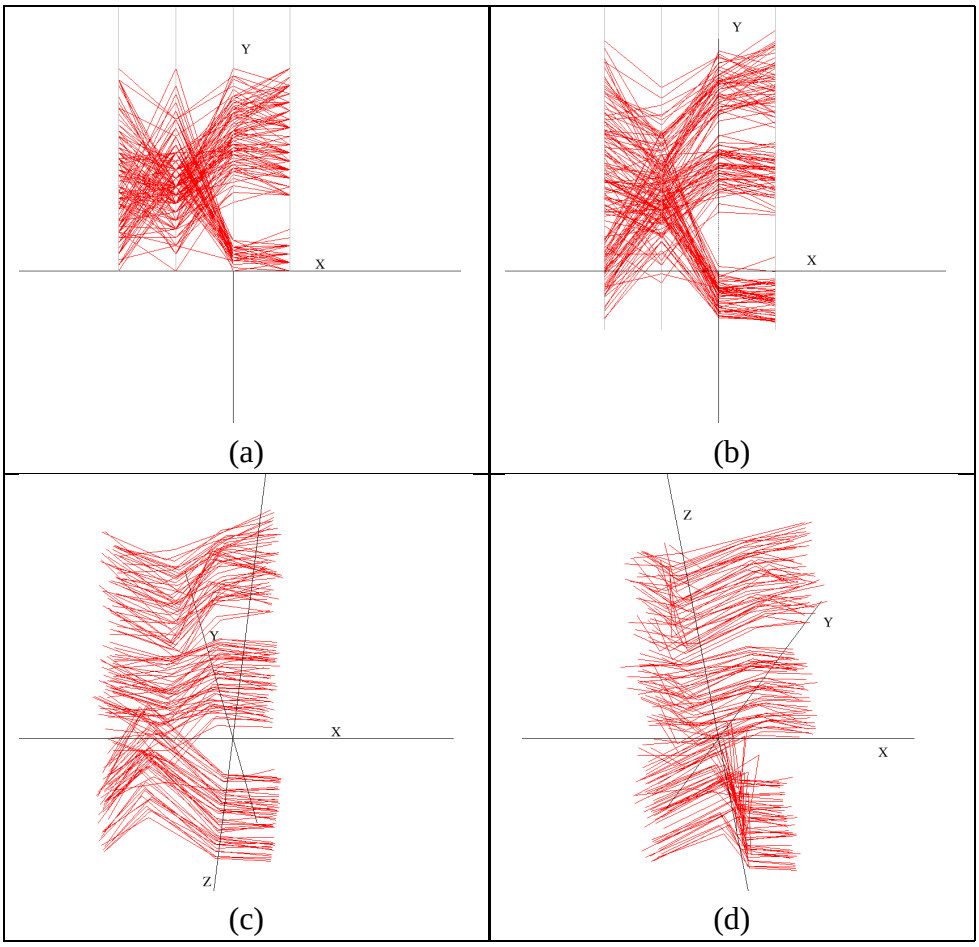


Figure 8.1: Visualizing the 4D multivariate Iris Data using 3D Parallel Coordinates with sorting of observation. Picture (a) shows the initial view, which is similar to the regular 2D parallel coordinates representation. Picture (b) through (d) shows the 3D parallel coordinates after various rotations have been applied – notice how the three clusters are now clearly separated in contrast to the image in Picture (a), where only one cluster is clearly visible.

inspections conducted throughout the three case studies presented in the evaluation chapter. Certainly we have not provided a definitive solution but rather a promising starting point, and, as such, it is open for improvement – an evolutionary process with alternating stages of evaluation and development.

Based on the results from the Appraisal chapter we have identified the next directions this research may pursue. These research avenues can be organized into practical and conceptual levels, and they are outlined below:-

PRACTICAL

- To allow logical operations between filters. The paradigm adopted by HyperCell is one of using one or various filters to support the exploration of the high-dimensional

datasets. So a natural step is to enable users to perform logical operations between filters. This means, for instance, to apply union and intersection operations between distinct *n*-dimensional windows.

The need for this ability was mainly identified during the work with the CMS case study. Instructors wanted to be able to create various brushing objects to 'isolate' individual students and their data and then logically (AND operation) combine them into a single visualization for comparison purpose.

- To develop modules for the *data analysis* part of the framework. This would involve, for example, providing automatic methods to find *extrema* (motivated during the case study involving the visualization of the 4D Rosenbrock function); to incorporate Multidimensional Scaling algorithms; or, simply to implement various re-ordering of variables or observations to bring out relationships between them.

  The latter has been tested as a prototype in association with a three-dimensional representation for parallel coordinates. The result is a new visualization technique that can be described by our framework as shown in Figure 8.2.

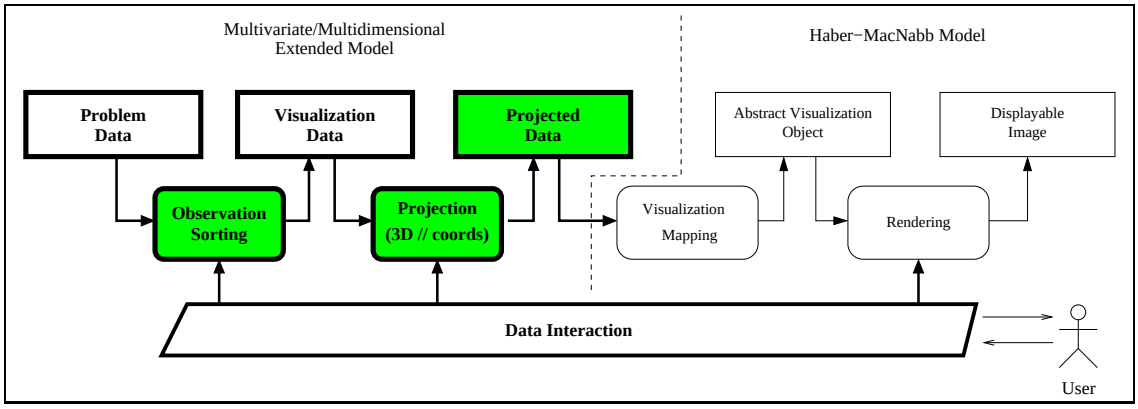


Figure 8.2: The 3D Parallel Coordinates technique described in terms of our suggested high-dimensional reference model. Represented in green are the components of the reference model adapted to describe the 3D Parallel Coordinates technique.

- To afford direct manipulation of the visual representation through the Render window. Direct manipulation in *visual space* is an important aspect regarding the *data interaction* stage of our visualization technique. This need for this feature has been identified at the design evaluation phase, as well as when *HyperCell* was applied to the case studies involving multivariate datasets.

  User interaction at this level requires some investigation towards effective and intuitive interaction mechanisms within 3D spaces, if we consider the 3D cells that can

be generated. At the current stage this would require some modification/refinement being done to the IRIS Explorer Render module to afford the mentioned interactive features.

- To experiment with different projections onto the 'plane' through the focus point (for multivariate data). For example, we could try alternative projection methods (e.g. perspective projection, or distorted projections) instead of the current orthogonal projection (i.e. ignoring the coordinates of variates not in the cell) and evaluate whether this would promote any gain of new information; or we could use, say, Euclidean (L_2) metric to measure the distance from the projection 'plane' determined by the location of the focus point to the observations in *n*-space – in this case we would be 'encoding' more information into the visualization, because the calculation of distance would involve *all* variates, including the ones that are not part of the cell.

CONCEPTUAL

- To establish the cognitive limits to the number of variables and the number of observations (for the multivariate case), possibly through some controlled experiment involving users. This is somehow a challenging enterprise, considering that it is difficult to define general tasks that could be used for that matter. One may argue that for a certain task – for example, to find correlation between pair of dimensions – 30 variates is the limit; whereas for the investigation of a multidimensional function around an *extrema* point this limit may be much lower because of cognitive constraints.

- To refine the framework by providing more detailed sub-categories for the data analysis stage. This would involve a more comprehensive study of the existing techniques for high-dimensional data, trying to identify common features that might lead to a clear definition of sub-categories for the *data analysis*.

- To evaluate the impact of the suggested lay-out strategies during the investigation process. Again this would involve more controlled experiments involving users. The aim here would be to compare two distinct situations: letting the user arrange the cells on their desktop at will, compared to a situation in which an arrangement is suggested, namely the *building* or the *fruit machine* lay-outs.

We have noticed during the evaluation stage that, for example, using a pre-arranged lay-out (namely the 'fruit machine') was slightly better in understanding the behaviour of the 4D Rosenbrock function around the minimum in comparison with using only one view; in contrast using a single view was 'easier' to follow when the focus points was moved about in n-space in comparison with having to look at several views organized in a lay-out.

- To understand and evaluate navigation strategies. Another evaluation procedure highly dependent on the task at hand. A good starting point towards devising efficient structures to support navigation has been described by Spence in his work on 'sensitive encoding' [183]. The author suggests several guidelines to build navigation mechanisms whose objective is the formation of a mental model, movement towards an objective or both.

8.4 In Conclusion

The methodology presented in this thesis is relevant not only to the particular problem domain of visualizing high-dimensional data, but to both fields of scientific and information visualization as a whole – it reinforces the relevance of investigating more formal approaches, such as reference models, towards a comprehensive understanding of how visualization can be useful for human activity.

The suggested framework serves as a model for subsequent work regarding multivariate and multidimensional data visualization, a) working as a device for comparison of methods; b) encouraging the development of new techniques; and, c) functioning as a foundation that will help the integration of structures, algorithms, and methods originally designed for either SciVis or InfoVis – thus encouraging the re-cycling rather than re-invention of already existing and well tested components. This is an important asset, especially if we consider that scientific and information visualization problems involving several variables and/or many dimensions are becoming increasingly more common.

Finally we believe this work has contributed to the principles and practice of visualization by describing a method to deal with high-dimensional applications that supports discovery, encourages exploration and, most importantly, fosters insight.

Bibliography

[1] C. Ahlberg and B. Shneiderman. Visual information seeking: tight coupling of dynamic query filters with starfield displays. In B. Adelson, S. Dumais, and J. S. Olson, editors, *Proceedings of the 1994 SIGCHI Conference on Human factors in computing systems*, volume 1, pages 313–317. ACM Press, 1994.

[2] C. Ahlberg, C. Williamson, and B. Shneiderman. Dynamic queries for information exploration: an implementation and evaluation. In *Proceedings of the 1992 SIGCHI Conference on Human factors in computing systems*, pages 619–626. ACM Press, 1992.

[3] C. Ahlberg and E. Wistrand. Ivee: an environment for automatic creation of dynamic queries applications. In *Conference companion on Human factors in computing systems*, pages 15–16. ACM Press, 1995.

[4] B. Alpern and L. Carter. The hyperbox. In G. M. Nielson and L. Rosenblum, editors, *Proceedings of the 2nd IEEE Conference on Visualization (VIS '91)*, pages 133–139, 418. IEEE Computer Society Press, 1991.

[5] D. F. Andrews. Plots of high-dimensional data. *Biometrics*, 29:125–136, 1972.

[6] K. Andrews. Case study: Visualising cyberspace: information visualisation in the harmony internet browser. In *Proceedings of the 1995 IEEE Symposium on Information Visualization*, page 97. IEEE Computer Society, 1995.

[7] M. Ankerst, S. Berchtold, and D. A. Keim. Similarity clustering of dimensions for an enhanced visualization of multidimensional data. In *Proceedings of the 1998 IEEE Symposium on Information Visualization*, pages 52–60, 153. IEEE Computer Society, 1998.

[8] M. Ankerst, D. Keim, and H.-P. Kriegel. Circle segments: A technique for visually exploring large multidimensional data sets. In R. Yagel and G. M. Nielson, editors,

Proceedings of the 7th IEEE Conference on Visualization (VIS '96), Hot Topic, New York, NY, USA, 1996. IEEE Computer Society and ACM Press.

[9] D. Asimov. The grand tour: A tool for viewing multidimensional data. *SIAM Journal on Scientific and Statistical Computing*, 6(1):128–143, 1985.

[10] C. L. Bajaj, V. Pascucci, G. Rabbiolo, and D. R. Schikore. Hypervolume visualization: a challenge in simplicity. In *Proceedings of IEEE/ACM Symposium on Volume Visualization*, pages 95–102,172. IEEE Computer Society Press, 1998.

[11] T. F. Banchoff. *Beyond the Third Dimension*. Scientific American Library Series. W.H. Freeman & Company, New York, NY, third edition, 1996.

[12] A. Bartkowiak and A. Szustalewicz. The grand tour as a method for detecting multivariate outliers. *Machine Graphics & Vision*, 6:487–505, 1997.

[13] W. Basalaj. Proximity visualization of abstract data. Technical Report 509, University of Cambridge Computer Laboratory, January 2001. url:http://www.pavis.org/essay/index.html.

[14] B. G. Becker. Research report: Volume rendering for relational data. In *Proceedings of the 1997 IEEE Symposium on Information Visualization*, pages 87–91. IEEE Computer Society Press, 1997.

[15] R. Becker and W. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, May 1987.

[16] R. A. Becker, S. G. Eick, and A. R. Wilks. Visualizing network data. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):16–28, 1995.

[17] J. Beddow. Shape coding of multidimensional data on a microcomputer display. In A. Kaufman, editor, *Proceedings of the First IEEE Conference on Visualization (VIS '90)*, pages 238–246, 478. IEEE Computer Society Press, 1990.

[18] R. Bellman. *Adaptative Control Processes: A Guided Tour*. Princeton University Press, 1961.

[19] C. Beshers and S. Feiner. Automated design of virtual worlds for visualizing multivariate relations. In *Proceedings of the 3rd IEEE Conference on Visualization (VIS '92)*, pages 283–290. IEEE Computer Society Press, 1992.

[20] C. Beshers and S. Feiner. Autovisual: rule-based design of interactive multivariate visualizations. *IEEE Computer Graphics and Applications*, 13(4):41–49, 1993.

[21] P. Bragatto, N. Mazzino, and P. Palamidese. Animated visualization of scalar fields. In G. Hegron and D. Thalmann, editors, *Computer Animation and Simulation '91*, Eurographics Technical Report Series, pages 115–127, 1991. Proceedings of the Eurographics Workshop in Vienna, Austria, September 1–2, 1991.

[22] R. K. Brath. Effective information visualization: Guidelines and metrics for 3d interactive representations of business data. Master's thesis, Computer Science Department - University of Toronto, 1999.

[23] K. Brodlie. A classification scheme for scientific visualization. In R. E. Earnshaw and D. Watson, editors, *Animation and scientific visualization - tools and applications*, chapter 8, pages 125–140. Academic Press Ltd, 1993.

[24] K. W. Brodlie. A new direction set method for unconstrained minimization without evaluating derivatives. *J. Inst. Maths Applics*, 15:385–396, 1975.

[25] K. W. Brodlie, L. Carpenter, R. A. Earnshaw, J. R. Gallop, R. J. Hubbold, A. M. Mumford, C. D. Osland, and P. Quarendon. *Scientific visualization: techniques and applications*. Springer-Verlag New York, Inc., 1992.

[26] Ken Brodlie, Andrew Poon, Helen Wright, Lesley Brankin, Greg Banecki, and Alan Gay. GRASPARC: a problem solving environment integrating computation and visualization. In D. Bergeron and G. Nielson, editors, *Proceedings of the 4th IEEE Conference on Visualization (VIS '93)*, pages 102–109, 1993.

[27] A. Buja and D. Asimov. Grand tour methods: An outline. In D. M. Allen, editor, *Proceedings of the 17th Symposium on the Interface Between Computing Science and Statistics*, pages 63–67. Elsevier, 1986.

[28] A. Buja, B. Cook, D. Asimov, and D. Hurley. Theory and computational methods for dynamic projections in high-dimensional data visualization, 1996.

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

[30] A. Buja, J. A. McDonald, J. Michalak, and W. Stuetzle. Interactive data visualization using focusing and linking. In *Proceedings of the 2nd IEEE Conference on Visualization (VIS '91)*, pages 156–163, 419. IEEE Computer Society Press, 1991.

[31] M. Büscher, P. Mogensen, and D. Shapiro. Spaces of practice. In W. Prinz et al., editor, *Proceedings of the Seventh European Conference on Computer Supported Cooperative Work*, pages 139–158. Kluwer Academic Publishers, 2001.

[32] S. Card, J. Mackinlay, and B. Shneiderman, editors. *Readings in Information Visualization - Using Vision to Think.* Morgan Kaufmann Publishers, Inc., 1999.

[33] S. K. Card and J. Mackinlay. The structure of the information visualization design space. In *Proceedings of the 1997 IEEE Symposium on Information Visualization*, pages 92–99. IEEE Computer Society Press, 1997.

[34] S. K. Card, J. D. Mackinlay, and B. Shneiderman, editors. *Readings in Information Visualization - Using Vision to Think*, chapter 1: Information Visualization, pages 1–34. Morgan Kaufmann Publishers, Inc., 1999.

[35] S. K. Card, G. G. Robertson, and W. York. The webbook and the webforager: An information workspace for the world-wide web. In *Proceedings of the 1996 SIGCHI Conference on Human Factors in Computing Systems*, pages 111–117. ACM Press, 1996.

[36] M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia. IEEE computer graphics and applications, special issue on information visualization. *IEEE Journal Press*, 17(4):42–51, July 1997.

[37] M. A. Carreira-Perpiñán. A review of dimension reduction techniques. Technical report cs-96-09, Dept. of Computer Science, University of Sheffield, 1996.

[38] M. Chalmers. Tutorial: Design and perception in information visualisation. In *25th International Conference on Very Large Data Bases*, 1999.

[39] J. M. Chambers, W. S. Cleveland, B. Kleiner, and P.A. Tukey. *Graphical Methods for Data Analysis*. Chapman and Hall, New York, 1976.

[40] J. X. Chen and S. Wang. Data visualization: Parallel coordinates and dimension reduction. *Computing in Science & Engineering*, 3(5):110–113, September-October 2001.

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

[42] E. H. Chi. *A Framework for Visualizing Information*, volume 1 of *Human-computer Interaction Series*. Kluwer Academic Publishers, 2002.

[43] E. H. Chi and J. Riedl. An operator interaction framework for visualization systems. In *Proceedings of the 1998 IEEE Symposium on Information Visualization*, pages 63–70. IEEE Computer Society, 1998.

[44] E. H. H. Chi, P. Barry, J. Riedl, and J. Konstan. A spreadsheet approach to information visualization. In *Proceedings of the 1997 IEEE Symposium on Information Visualization*, page 17. IEEE Computer Society, 1997.

[45] R. Chimera. Value bars: an information visualization and navigation tool for multi-attribute listings. In *Proceedings of the 1992 SIGCHI Conference on Human factors in computing systems*, pages 293–294. ACM Press, 1992.

[46] M. C. Chuah, S. F. Roth, J. Mattis, and J. Kolojejchick. SDM: malleable information graphics. In *Proceedings of the 1995 IEEE Symposium on Information Visualization*, page 36. IEEE Computer Society, 1995.

[47] W. S. Cleveland. *The elements of graphing data*. Wadsworth Advanced Books and Software, 1985.

[48] W. S. Cleveland. *Visualizing Data*. Hobart Press, AT&T Bell Laboratories, NJ, USA, 1993.

[49] W. S. Cleveland and M. E. McGill, editors. *Dynamic Graphics for Statistics*. Statistics/probability series. Wadsworth and Brooks/Cole Inc., 1988.

[50] W. S. Cleveland, M. E. McGill, and R. McGill. The shape parameter of a two-variable graph. *Journal of the American Statistical Association*, 83:289–300, 1988.

[51] W. G. Cochran. *Sampling Techniques*. Wiley Series in Probability and Statistics. Wiley Text Books, third edition, July 1977.

[52] M. Cohen and K. Brodlie. Focus and context for volume visualization. In *Proceedings Theory and Practice of Computer Graphics 2004*, pages 32–39. IEEE Computer Society Press, 2004.

[53] M. B. Cohen, C. J. Colbourn, L. A. Ives, and A. C. H. Ling. Kirkman triple systems of orders 21 and 27. *Mathematics of Computation*, 71(238):873–881, November 2002.

[54] B. N. Collins. Data visualization - has it all been seen before? In R. A. Earnshaw and D. Watson, editors, *Animation and Scientific Visualization - Tools & Applications*, pages 3–28. Academic Press, 1993.

[55] D. Cook, A. Buja, J. Cabrera, and H. Hurley. Grand tour and projection pursuit. *Journal of Computational and Graphical Statistics*, 4(3):155–172, 1995.

[56] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling,*. Chapman & Hall,, London, 1994.

[57] S. L. Crawford and T. C. Fall. *Visualization in Scientific Computing*, chapter Projection Pursuit Techniques for the Visualization of High Dimensional Datasets, pages 94–108. IEEE Computer Society Press, 1990.

[58] R. Curtis and S. Scarfone. XFace, an X tool for presenting multivariate data, and its use with software metrics. In *Proceedings of the Eleventh Annual IEEE International Phoenix Conference on Computers and Communications*, pages 525 – 530. IEEE Computer Society Press, 1992.

[59] T. A. DeFanti, M. D. Brown, and B. H. McCormick. Visualization: Expanding scientific and engineering research opportunities. *Computer*, 22(8):12–16, 22–25, 1989.

[60] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollisand. *Graph drawing: algorithms for the visualization of graphs*. Pearson Education, first edition, 1998.

[61] P. Diaconis and J. H. Friedman. M and n plots. In H. Rizvi, J. Rustagi, and D. Siegmund, editors, *Recent Advances in Statistics: Papers in Honor of Herman Chernoff on his Sixtieth Birthday*, pages 425–447. Academic Press, 1983.

[62] D. Duke. Modular techniques in information visualization. In Peter Eades and Tim Pattison, editors, *Australian Symposium on Information Visualisation, (invis.au 2001)*, volume 9 of *Conferences in Research and Practice in Information Technology*, pages 11–18. ACS, 2001.

[63] M. Egenhofer and J. Richards. Exploratory access to geographic data based on the map-overlay metaphor. *Journal of Visual Languages and Computing*, 4(2):105–125, 1993.

[64] S. G. Eick. Data visualization sliders. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 119–120. ACM Press, 1994.

[65] Stephen G. Eick, Joseph L. Steffen, and Eric E. Sumner, Jr. SeeSoft-a tool for visualizing line oriented software statistics. *IEEE Trans. Softw. Eng.*, 18(11):957–968, 1992.

[66] K. M. Fairchild, S. E. Poltrock, and G. W. Furnas. Semnet: Three-dimensional representations of large knowledge bases. In R. Guindon, editor, *Cognitive Science and its Applications for Human-Computer Interaction*, pages 201–233, Hillsdale, NJ, USA, 1988. Lawrence Erlbaum.

[67] S. K. Feiner and C. Beshers. Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds. In *Proceedings of the Third Annual ACM SIGGRAPH Symposium on User Interface Software and Technology*, pages 76–83. ACM Press, 1990.

[68] R. A. Fisher. The use of multiple measurements in axonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[69] M. A. Fisherkeller, J. H. Friedman, and J. W. Tukey. Prim9, an interactive multi-dimensional data display and analysis system. In *Dynamic Graphics for Statistics*, Statistics/Probability Series, pages 91–109. The Wadsworth & Brooks/Cole, 1975.

[70] K. Fishkin and M. C. Stone. Enhanced dynamic queries via movable filters. In I. R. Katz, R. Mack, L. Marks, M. B. Rosson, and J. Nielsen, editors, *Proceedings of the 1995 SIGCHI Conference on Human factors in computing systems*, pages 415–420. ACM Press/Addison-Wesley Publishing Co., 1995.

[71] Institute for Manufacturing Department of Engineering. Decision support tools - polar chart. url:http://www.ifm.eng.cam.ac.uk/dstools/represent/polar.html, 2004.

[72] J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, 23:881–889, 1977.

[73] Ying-Huey Fua, M. O. Ward, and E. A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. In *Proceedings of the 10th IEEE Conference on Visualization (VIS '99)*, pages 43–50, 508. IEEE Computer Society, 1999.

[74] Ying-Huey Fua, M. O. Ward, and E. A. Rundensteiner. Structure-based brushes: a mechanism for navigating hierarchically organized data and information spaces. *IEEE Transactions on Visualization and Computer Graphics*, pages 150–159, 2000.

[75] G. Furnas. Generalized fisheye views. In M. Mantei and P. Orbeton, editors, *Proceedings of the 1986 SIGCHI Conference on Human factors in computing systems*, pages 16–23, New York, NY, USA, 1986. ACM Press.

[76] G. Furnas and A. Buja. Prosections views: Dimensional interface through sections and projections. *Journal of Computational and Graphical Statistics*, 3(4):323–385, 1994.

[77] G. W. Furnas and B. B. Bederson. Space scale diagrams: Understanding multiscale interfaces. In I. R. Katz, R. Mack, L. Marks, M. B. Rosson, and J. Nielsen, editors, *Proceedings of the 1995 SIGCHI Conference on Human factors in computing systems*, pages 234–241. ACM Press/Addison-Wesley Publishing Co., 1995.

[78] G. Geisler. Making information more accessible: A survey of information visualization applications and techniques. http://www.ils.unc.edu/geisg/info/infovis/paper.html, February 1998.

[79] N. Gershon and S. G. Eick. Information visualisation. *IEEE Computer Graphics Application*, 17(4):29–31, 1997.

[80] J. Goldstein and S. F. Roth. Using aggregation and dynamic queries for exploring large data sets. In B. Adelson, S. Dumais, and J. S. Olson, editors, *Proceedings of the 1994 SIGCHI Conference on Human factors in computing systems*, volume 1, pages 23–29. ACM Press, 1994.

[81] M. Graham and J. Kennedy. Using curves to enhance parallel coordinate visualisations. In *Proceedings of the Seventh International Conference on Information Visualization*, pages 10–16. IEEE Computer Society, 2003.

[82] M. Green. Toward a perceptual science of multidimensional data visualization: Bertin and beyond. url: http://www.ergogero.com/dataviz/dviz0.html, 1998. Copyright by ERGO/GERO.

[83] G. Grinstein, A. Inselberg, and S. Laskowski. Key problems and thorny issues in multidimensional visualization. In *Proceedings of the 9th IEEE Conference on Visualization (VIS '98)*, pages 505–506. IEEE Computer Society Press, 1998.

[84] G. Grinstein, T. Mihalisin, H. Hinterberger, and A. Inselberg. Visualizing multidimensional (multivariate) data and relations. In *Proceedings of the 5th IEEE Conference on Visualization (VIS '94)*, pages 404–409. IEEE Computer Society Press, 1994.

[85] G. Grinstein, R. M. Pickett, and M. G. Williams. Exvis: An explanatory visualization environment. *Graphics Interface*, pages 254–261, 1989.

[86] G. Grinstein, M. Trutschl, and U. Cvek. High-dimensional visualizations. In *Proceedings of The Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2001)*, 2001.

[87] C. Plaisant H. P. Kumar and B. Shneiderman. Browsing hierarchical data with multilevel dynamic queries and pruning. *International Journal of Human-Computer Studies*, 46(1):103–124, January 1997.

[88] R. B. Haber and D. A. McNabb. Visualization idioms: A conceptual model for scientific visualization systems. In Gregory M. Nielson, B. Shriver, and L. J. Rosenblum, editors, *Visualization in Scientific Computing*, pages 74–93. IEEE Comp. Society Press, 1990.

[89] N. Hambly, M. Read, R. Mann, E. Sutorius, I. Bond, H. MacGillivray, P. Williams, and A. Lawrence. The SuperCOSMOS science archive. In F. Ochsenbein, M. Allen, and D. Egret, editors, *Proceedings of the Astronomical Data Analysis Software and Systems (ADASS) XIII*, volume 314, pages 137–140, 2004.

[90] A. J. Hanson. *Graphics Gems V*, chapter Rotations for N-Dimensional Graphics, pages 55–64. Academic Press, 1995.

[91] D. Helic, H. Maurer, and N. Scherbakov. Web based training: What do we expect from the system. In S.S. Young, J. Greer, H. Maurer, and Y. S. Chee, editors, *Proceedings of Eighth International Conference on Computers in Education/International Conference on Computer-Assisted Instruction (ICCE/ICCAI 2000)*, pages 222–230. AACE-APC/National Tsing Hua University, 2000.

[92] R. J. Hendley, N. S. Drew, A. M. Wood, and R. Beale. Case study: Narcissus: visualising information. In *Proceedings of the 1995 IEEE Symposium on Information Visualization*, page 90. IEEE Computer Society, 1995.

[93] B. Hibbard. Top ten visualization problems. *SIGGRAPH Computer Graphics Newsletter - VisFile*, 33(2):21–22, 1999.

[94] P. Hoffman, G. Grinstein, K. Marx, L. Grosse, and E. Stanley. DNA visual and analytic data mining. In *Proceedings of the 8th IEEE Conference on Visualization (VIS '97)*, pages 437–441, Ney York, NY, USA, 1997. ACM Press and IEEE Computer Society Press.

[95] D. Howe. The free on-line dictionary of computing. url:http://www.foldoc.org/, 1993.

[96] P. J. Huber. Projection pursuit. *Annals of Statistics*, 13(2):435–475, June 1985.

[97] A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multi-dimensional geometry. In A. Kaufman, editor, *Proceedings of the First IEEE Conference on Visualization (VIS '90)*, pages 361–370. IEEE Computer Society Press, 1990.

[98] V. Interrante. Harnessing natural textures for multi-variate visualization. *IEEE Computer Graphics and Applications*, 20(6):6–11, 2000.

[99] J. E. Jackson. *A user's guide to principal components*. Wiley series in probability and mathematical statistics. John Wiley & Sons, Inc., 1991.

[100] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall Inc., New Jersey, 1988.

[101] T. J. Jankun-Helly and K.-L. M. Visualization exploration and encapsulation via a spreadsheet-like interface. *IEEE Transaction on Visualization and Computer Graphics*, 7(3):275–287, July-September 2001.

[102] D. F. Jerding and J. T. Stasko. The information mural: a technique for displaying and navigating large information spaces. In *Proceedings of the 1995 IEEE Symposium on Information Visualization*, page 43. IEEE Computer Society, 1995.

[103] M. C. Jones and R. Sibson. What is projection pursuit? *Journal of the Royal Statistical Society. Series A (General)*, 150(1):1–37, 1987.

[104] D. A. Henderson Jr. and S. K. Card. Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Transactions on Graphics (TOG)*, 5(3):211–243, July 1986.

[105] E. Kandogan. Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. In *Proceedings of the IEEE Information Visualization Symposium, Hot Topics,*, pages 4–8, 2000.

[106] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, New York, USA, 1990.

[107] D. Keim, H.-P Kriegel, and M. Ankerst. Recursive pattern: a technique for visualizing very large amounts of data. In *Proceedings of the 6th IEEE Conference on Visualization (VIS '95)*, pages 279–286, 463. IEEE Computer Society Press, 1995.

[108] D. A. Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):59–78, January-March 2000.

[109] D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 7(1):100–107, January-March 2002.

[110] D. A. Keim and H. Krigel. VisDB: Database exploration using multidimensional visualization. *IEEE Computer Graphics and Applications*, 14(5):40–49, September-October 1994.

[111] P. R. Keller and M. M. Keller. *Visual Cues, Practical Data Visualization*. IEEE Press, 1993.

[112] T. P. Kirkman. Note on an unanswered proze question. *Cambridge and Dublin Math. J.*, 5:255–262, 1850.

[113] B. Kleiner and J. A. Hartigan. Representing points in many dimensions by trees and castles. *Journal of the American Statistical Association (in Applications)*, 76(374):260–269, 276, 1981.

[114] T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, New York, NY, USA, third extended edition edition, 2001.

[115] A. Konig. Dimensionality reduction techniques for multivariate data classification, interactive visualization, and analysis-systematic feature selection vs. extraction. In R. J. Howlett and L. C. Jain, editors, *Proceedings of the Fourth International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies*, volume 1, pages 44–55. University of Brighton, IEEE Press, 2000.

[116] R. R. Korfhage. To see, or not to see - is that the query? In *Proceedings of the 14th annual international ACM SIGIR Conference on Research and development in information retrieval*, pages 134–141. ACM Press, 1991.

[117] R. Kosara, G. Sahling, and H. Hauser. Linking scientific and information visualization with interactive 3d scatterplots. In *Proceedings of the 12th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 133–140, 2004.

[118] J. B. Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrica*, 29:115–129, 1964.

[119] E. LaMar, B. Hamann, and K. I. Joy. A magnification lens for interactive volume visualization. In *Proceedings of the Ninth Pacific Conference of Computer Graphics and Applications*, pages 223–232, 2001.

[120] J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In I. R. Katz, R. Mack, L. Marks, M. B. Rosson, and J. Nielsen, editors, *Proceedings of the 1995 SIGCHI Conference on Human factors in computing systems*, pages 401–408. ACM Press/Addison-Wesley Publishing Co., 1995.

[121] R. Laurini and D. Thompson. *Fundamentals of Spatial Information Systems*. Academic Press, New York, NY, USA, 1992.

[122] J. LeBlanc, M. O. Ward, and N. Wittels. Exploring n-dimensional databases. In A. Kaufman, editor, *Proceedings of the First IEEE Conference on Visualization (VIS '90)*, pages 230–237. IEEE Computer Society Press, 1990.

[123] J. Leftwich. Infospace: A conceptual method for interacting with information in a three-dimensional virtual environment. In *Proceedings of Third International Conference on Cyberspace*, 1993. On line version at: http://www.well.com/user/jleft/orbit/infospace/.

[124] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transaction on Computer-Human Interaction*, 1(2):126–160, June 1994.

[125] H. Levkowitz. Color icons-merging color and texture perception for integrated visualization of multiple parameters. In G. M. Nielson and L. Rosenblum, editors, *Proceedings of the 2nd IEEE Conference on Visualization (VIS '91)*, pages 164–170, 420. IEEE Computer Society Press, 1991.

[126] H. Lohninger. Inspect, a program system to visualize and interpret chemical data. *Chemometrics and Intelligent Laboratory Systems*, 22:147–153, 1994.

[127] The Numerical Algorithms Group Ltd. Iris explorer module writer's guide. url:http://www.nag.co.uk/visual/IE/iecbb/DOC/html/unix-ie5-0.htm.

[128] K.-L. Ma. Visualizing visualization - user interfaces for managing and exploring scientific data. *Computer Graphics and Application*, 20(5):16–19, Sep/Oct 2000.

[129] A. M. MacEachren. *How Maps Work, Representation, Visualization and Design*. The Guildford Press, 1995.

[130] Macromedia. Macromedia director mx 2004. url:http://www.macromedia.com/, 2004.

[131] Wendy Martinez and Angel Martinez. *Computational Statistics Handbook with MATLAB.* Chapman and Hall, 2002.

[132] R. Mazza. Diagnosis the state of the student's knowledge in a web-based learning environment. In *Proceedings of the 4th International Conference on New Educational Environments Lugano*, pages 8–11, 2002.

[133] R. Mazza and V. Dimitrova. Coursevis: Externalising student information to facilitate instructors in distance learning. In U. Hoppe, F. Verdejo, and J. Kay, editors, *Proceedings of the International Conference in Artificial Intelligence in Education - (AIED 2003)*, pages 279–286. IOS press, 2003.

[134] R. Mazza and V. Dimitrova. Informing the design of a course data visualisator: an empirical study. In *Proceedings of the 5th International Conference on New Educational Environments -(ICNEE 2003)*, pages 215–220, 2003.

[135] R. Mazza and V. Dimitrova. Visualising student tracking data to support instructors in web-based distance education. In *Proceedings of the 13th International World Wide Web Conference (WWW 2004) - Educational Track*, pages 17–22, 2004.

[136] B. H. McCormick, T. A. DeFanti, and M. D. Brown. Visualization in scientific computing. *ACM Computer Graphics*, 21(6):1–14, 1987. Special Issue.

[137] J. A. McDonald and W. Stuetzle. Painting multiple views of complex objects. In *Proceedings of the European Conference on Object Oriented Programming (ECOOP/OOPSLA '90)*, pages 245–257. ACM Press, 1990.

[138] A. Mead. Review of the development of multidimensional scaling methods. *Statistician*, 41(1):27–39, 1992.

[139] J. D. Mickinlay, G. G. Robertson, and S. K. Card. The perspective wall: detail and context smoothly integrated. In J. M. Carroll and P. P. Tanner, editors, *Proceedings of the 1987 SIGCHI Conference on Human Factors in Computing Systems and Graphics Interface*, pages 173–179, New York, NY, USA, 1987. ACM Press.

[140] T. Mihalisin, E. Gawlinski, J. Timlin, and J. Schwegler. Visualizing a scalar field on an n-dimensional lattice. In A. Kaufman, editor, *Proceedings of the First IEEE Conference on Visualization (VIS '90)*, pages 255–262 and 479–480. IEEE Computer Society Press, 1990.

[141] T. Mihalisin, J. Timlin, and J. Schwegler. Visualizing multivariate functions, data, and distributions. *IEEE Computer Graphics and Applications*, 11(3):28–35, 1991.

[142] M. C. Minnotte and R. W. West. The data image: a tool for exploring high dimensional data sets. In *Proceedings of the ASA Section on Statistical Graphics*, pages 0–0, 1998. in press.

[143] NAG IRIS Explorer. Web site, 2003. http://www.nag.co.uk.

[144] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7(4):308–313, January 1965.

[145] C. M. Newton. Graphics: From alpha to omega in data analysis. In P. C. C. Wang, editor, *Proceedings of the Symposium on Graphical Representation of Multivariate Data*, pages 59–92, New York, NY, USA, 1978. Academic Press.

[146] J. Nielsen. *Hypertext and Hypermedia*. Academic Press, 1990.

[147] L. Nigay and F. Vernier. Design method of interaction techniques for large information spaces. In *Proceedings of the working Conference on Advanced visual interfaces*, pages 37–46. ACM Press, 1998.

[148] M. A. Noll. A computer technique for displaying n-dimensional hyperobjects. *Communications of the ACM*, 10(8):469–473, 1967.

[149] C. North and B. Shneiderman. A taxonomy of multiple window coordinations. Technical report, University of Maryland, College Park, Dept. of Computer Science, 1997.

[150] C. North and B. Shneiderman. Snap-together visualization: A user interface for coordinating visualizations via relational schemata. In *Proceedings of Advanced Visual Interfaces 2000*, pages 128–135, 2000.

[151] C. North, B. Shneiderman, and C. Plaisant. User controlled overviews of an image library: a case study of the visible human. In *Proceedings of the first ACM international Conference on Digital libraries*, pages 74–82. ACM Press, 1996.

[152] Open Visualization Data Explorer. Web site, 2003. http://www.opendx.org/.

[153] A. Osyczka. *Design Optimization*, chapter Multicriterion Optimization for Engineering Design, pages 193–227. Academic Press, 1985.

[154] Davis P. J. *Interpolation and Approximation*. Dover Publications, 1975.

[155] R. S. Palais. The visualization of mathematics: Towards a mathematical exploratorium. *Notices of the American Mathematical Society*, 46(6):647–658, June/July 1999.

[156] R. M. Pickett and G. G. Grinstein. Iconographic displays for visualizing multidimensional data. In *Proceedings of the 1988 IEEE International Conference on Systems, Man, and Cybernetics*, volume 1, pages 514–519. IEEE Computer Society Press, 1988.

[157] C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman. Lifelines: visualizing personal histories. In *Proceedings of the 1996 SIGCHI Conference on Human factors in computing systems*, pages 221–227. ACM Press, 1996.

[158] R. Rao and S. K. Card. The table lens: Merging graphical and symbolic representation in an interactive focus+context visualization for tabular information. In B. Adelson, S. Dumais, and J. S. Olson, editors, *Proceedings of the 1994 SIGCHI Conference on Human Factors in Computing Systems*, volume 1, pages 318–322 and 481–482. ACM Press, 1994.

[159] P. Rheingans and C. Landreth. *Perceptual Issues in Visualization*, chapter Perceptual principles for effective visualizations, pages 59–73. Springer, 1995.

[160] T-M. Rhyne, M. Tory, T. Munzner, M. Ward, C. Johnson, and D. H. Laidlaw. Panel session: Information and scientific visualization: Separate but equal or happy together at last. In *Proceedings of IEEE Visualization 2003*, 2003. url=http://www.cs.brown.edu/research/vis/docs/pdf/Rhyne-2003-ISV.pdf.

[161] J. Roberts, editor. *Coordinated and Multiple Views in Exploratory Visualization, 2003*, 2003.

[162] J. C. Roberts. Guest editor's introduction: special issue on coordinated and multiple views in exploratory visualization. *Information Visualization*, 2(4):199–200, December 2003.

[163] G. Robertson, S. Card, and J. Mackinlay. Information visualization using 3-d interactive animation. *Communications of the ACM*, 36(4):57–71, April 1993.

[164] G. G. Robertson and J. D. Mackinlay. The document lens. In *Proceedings of the 6th annual ACM symposium on User interface software and technology*, pages 101–108. ACM Press, 1993.

[165] G. G. Robertson, J. D. Mackinlay, and S. K. Card. Cone trees: animated 3d visualizations of hierarchical information. In *Proceedings of the 1991 SIGCHI Conference on Human factors in computing systems*, pages 189–194. ACM Press, 1991.

[166] G. E. Rosario, E. A. Rundensteiner, D. C. Brown, and M. O. Ward. Mapping nominal values to numbers for effective visualization. In T. Muzner and S. North, editors, *Proceedings of the 2003 IEEE Symposium on Information Visualization*, pages 113–120. IEEE Computer Society Press, 2003.

[167] L. Rosenblum, R. A. Earnshaw, J. Encarnacao, H. Hagen, A. Kaufman, S. Klimenko, F. Post G. Nielson, and D. Thalmann, editors. *Scientific Visualization - Advances and Challenges*. Academic Press Limited, 1994.

[168] H. H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, October 1959.

[169] G. Ross and M. Chalmers. A visual workspace for constructing hybrid multidimensional scaling algorithms and coordinating multiple views. *Information Visualization*, 2(4):247–257, December 2003.

[170] G. Salton, J. Allan, C. Buckley, and A. Singhal. Automatic analysis, theme generation, and summarization of machine-readable texts. In S. K. Card, J. D. Mackinlay, and B. Shneiderman, editors, *Readings in Information Visualization*, pages 413–418. Morgan Kaufmann Publishers, Inc., 1999.

[171] M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. In *Proceedings of the 1992 SIGCHI Conference on Human factors in computing systems*, pages 83–91. ACM Press, 1992.

[172] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit An Object-Oriented Approach To 3D Graphics*. Kitware Inc., 3rd edition, 2003. http://public.kitware.com/VTK/.

[173] G. H. Schut. Review of interpolation methods for digital terrain models. *The Canadian Surveyor*, 30(5):389–412, 1976.

[174] V. Sevastyanov. New method of visualization for multidimensional functions. url: http://www.multistat.com/download/Visualization.pdf, 2001. (Last accessed on 22 June, 2004).

[175] B. Shneiderman. Tree visualization with treemaps: a 2d space-filling approach. *ACM Transaction on graphics*, 11(1):92–99, 1992.

[176] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, pages 336–343. IEEE Computer Society, 1996.

[177] D. Shreiner, M. Woo, J. Neider, T. Davis, and D. Shreiner. *OpenGL Programming Guide*. Addison-Wesley Pub Co., fourth edition, November 2003.

[178] J. H. Siegel, E. J. Farrel, R. M. Goldwyn, and H. P. Friedman. The surgical implication of physiologic patterns in myocardial infarction shock. *Surgery*, 72:126–141, 1972.

[179] H. Siirtola. Direct manipulation of parallel coordinates. In *Proceedings of 2000 IEEE International Conference on Information Visualization*, pages 373–378, 2000.

[180] H. Sowizral, K. Rushforth, and M. Deering. *The Java 3D(TM) API Specification*. Addison-Wesley Pub Co., second edition, May 2000.

[181] R. Spence. *Information Visualization*. ACM Press Books, 2001.

[182] R. Spence. Rapid, serial and visual: a presentation technique with potential. *Information Visualization*, 1(1):13–19, March 2002.

[183] R. Spence. Sensitive encoding to support information space navigation: a design guideline. *Information Visualization*, 1:120–129, 2002.

[184] R. Spence and M. Apperley. Database navigation: an office environment for the professional. *Behaviour and Information Technology*, 1(1):34–54, 1982.

[185] A. Spoerri. Infocrystal: a visual tool for information retrieval & management. In *Proceedings of the second international Conference on Information and knowledge management (CIKM)*, pages 11–20. ACM Press, 1993.

[186] StartWright. Project management tools. url:http://www.startwright.com/project1.htm, 2004.

[187] M. C. Stone, K. Fishkin, and E. A. Bier. The movable filter as a user interface tool. In B. Adelson, S. Dumais, and J. S. Olson, editors, *Proceedings of the 1994 SIGCHI Conference on Human factors in computing systems*, volume 1, pages 306–312. ACM Press, 1994.

[188] T. Strothotte, editor. *Computational visualization: Graphics, abstraction, and interactivity*. Springer, 1998.

[189] D. Sunday. About lines and distance of a point to a line (2d & 3d). url=http://softsurfer.com/Archive/algorithm_0102/algorithm_0102.htm, February 2001.

[190] Leah A. Sutton. The principle of vicarious interaction in computer-mediated communications. *International Journal of Educational Telecommunications*, 7(3):223–242, 2001.

[191] D. F. Swayne, D. Cook, and A. Buja. Xgobi: Interactive dynamic data visualization in the x window system. *Journal of Computational and Graphical Statistics*, 7(1):113–130, 1998.

[192] M. Tory and T. Möller. A model-based visualization taxonomy. Technical Report CMPT-TR2002-06, Computing Science Department, Simon Fraser University, 2002.

[193] M. Tory and T. Möller. Human factors in visualization research. *IEEE Transactions on Visualization and Computer Graphics(TVCG)*, 10(1):72–84, January-February 2004.

[194] D. Tost, A. Puig, and M. Ferre. Visual clues in multimodal rendering. Technical Report LSI-02-38-R, Departament de Matemtica Aplicada i Anlisi - Universitat de Barcelona, May 22 2002.

[195] E. R. Tufte. *Envisioning Information*. Graphic Press, Cheshire, Connecticut, 1990.

[196] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphic Press, Cheshire, Connecticut, second edition, September 2001. (first edited in 1983).

[197] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley Press, 1977.

[198] L. Tweedie, R. Spence, H. Dawkes, and H. Su. Externalising abstract mathematical models. In *Proceedings of the 1996 SIGCHI Conference on Human factors in computing systems*, pages 406–ff. ACM Press, 1996.

[199] R. van Liere and J. J. van Wijk. Visualization of multi-dimensional scalar functions using hyperslice. *CWI Quarterly*, 7(2):147–158, 1994.

[200] J. J. van Wijk and R. van Liere. Hyperslice: visualization of scalar functions of many variables. In D. Bergeron and G. Nielson, editors, *Proceeding of the 4th IEEE Conference on Visualization (VIS '93)*, pages 119–125. IEEE Computer Society Press, 1993.

[201] A. Walestein. *Cognitive Support in Software Tools: a Distributed Cognition Framework*. PhD thesis, Computer Science Department, Simon Fraser University, Burnaby, B.C., Canada, 2002.

[202] N. Walliman and B. Baiche. *Your Research Project - a guide step-by-step for the first-time research*. SAGE Publications, 2001.

[203] J. Walton. Nag's iris explorer. In C. R. Johnson and C. D. Hansem, editors, *Visualization Handbook*. Academic Press, 2004. To appear. url=http://www.nag.co.uk/doc/TechRep/ietr.html#NP3654.

[204] M. Q. Wang Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for using multiple views in information visualization. In *Proceedings of the working Conference on advanced visual interfaces*, pages 110–119. ACM Press, 2000.

[205] M. O. Ward. Xmdvtool: integrating multiple methods for visualizing multivariate data. In *Proceedings of the 5th IEEE Conference on Visualization (VIS '94)*, pages 326–333. IEEE Computer Society Press, 1994.

[206] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers, 2000.

[207] WebCT. Web site, 2004. http://www.webct.com.

[208] R. Wegenkittl, H. Loffelmann, and E. Groller. Visualizing the behaviour of higher dimensional dynamical systems. In *Proceeding of the 8th IEEE Conference on Visualization (VIS '97)*, pages 119–125, 533, 1997.

[209] E. J. Wegman and Q. Luo. High dimensional clustering using parallel coordinates and the grand tour. Technical Report 124, Center for Computational Statistics, George Mason University, 1996.

[210] C. Williamson and B. Shneiderman. The dynamic homefinder: evaluating dynamic queries in a real-estate information exploration system. In *Proceedings of the 15th annual international ACM SIGIR Conference on research and development in information retrieval*, pages 338–346. ACM Press, 1992.

[211] J. A. Wise, J. J. Thomas, K. Pennock, D. Lantrip, M. Pottier, A. Schur, and V. Crow. Visualizing the non-visual: spatial analysis and interaction with information from text documents. In *Proceedings of the 1995 IEEE Symposium on Information Visualization*, page 51. IEEE Computer Society, 1995.

[212] P. C. Wong and R. D. Bergeron. *Scientific Visualization - Overviews, Methodologies and Techniques*, chapter 30 Years of Multidimensional Multivariate Visualization, pages 3–33. IEEE Computer Society Press, 1997.

[213] J. Wood, H. Wright, and K. Brodlie. Collaborative visualization. In *Proceedings of the 8th Conference on Visualization (VIS '97)*, pages 253–260. IEEE Computer Society Press, 1997.

[214] H. Wright, K. Brodlie, and T. David. Navigating high-dimensional spaces to support design steering. In T. Ertl, B. Hamann, and A. Varshney, editors, *Proceedings of IEEE Visualization 2000*, pages 291–296. IEEE Computer Society Press, 2000.

[215] H. Wright, K. Brodlie, J. Wood, and J. Procter. Problem solving environments: Extending the rôle of visualization systems. In A. Bode, T. Ludwig II, W. Karl, and R. Wismüller, editors, *Proceedings of the Euro-Par 2000 - Parallel Processing: 6th International Euro-Par Conference*, volume 1900 of *Lecture Notes in Computer Science*. Springer, 2000.

[216] W. Wright. Research report: information animation applications in the capital markets. In *Proceedings of the 1995 IEEE Symposium on Information Visualization*, page 19. IEEE Computer Society, 1995.

[217] J. Yang, W. Peng, M. O. Ward, and E. A. Rundensteiner. Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional

datasets. In T. Muzner and S. North, editors, *Proceedings of the 2003 IEEE Symposium on Information Visualization*, pages 105–112. IEEE Computer Society Press, 2003.

[218] J. Yang, M. O. Ward, E. A. Rundensteiner, and S. Huang. Visual hierarchical dimension reduction for exploration of high dimensional datasets. In G.-P. Bonneau, S. Hahmann, and C.D. Hansen, editors, *Proceedings of the Joint Eurographics/IEEE TVCG Symposium on Data Visualization 2003*, pages 19–28. IEEE Press/ACM Press, 2003.