Institute for Computer Graphics and Vision
Graz University of Technology
Graz

Visualization of Gene Expression Data with Advanced Scatterplots

Master's Thesis

Jürgen Pillhofer, BSc
juergen.pillhofer@student.tugraz.at

April 2010

Supervisor:
Univ.-Prof. Dipl.-Ing. Dr. techn. Dieter Schmalstieg
Advisor:
Dipl.-Ing. Marc Streit
Dipl.-Ing. Alexander Lex

Abstract

Caleydo is an information visualisation framework focusing on pathway exploration and the visualisation of gene expression data. In this thesis we provide another visualisation method for gene expression data based on scatterplots and scatterplot-matrices. While methods like dimension reduction techniques for this kind of visualisation have their own advantages, in our software implementation we focus on scatterplot matrices to present the data. Investigating modern InfoVis techniques is the key to provide a seamless integration of the scatterplot view into the framework. To maximize the usability, we combine many of these techniques to enhance a normal scatterplot into a powerful tool for the analysis of multivariate data. Using scatterplots matrices can produce a vast amount of data. Because high responsiveness in a real-time system is mandatory, methods which alleviate the performance issues are especially useful for our implementation.

Keywords: scatterplot, scatterplot matrix, focus plus context, details on demand, gene expression, Caleydo, OpenGL, JOGL

Zusammenfassung

Caleydo ist ein Informationsvisualisierung Framework, welches sich auf die Darstellung von Stoffwechselwegen und Genexpressionsdaten spezialisiert hat. In dieser Arbeit stellen wir eine weitere Visualisierungsmethode mit Scatterplots (Streudiagrammen) und Scatterplot Matrizen vor. Während Dimensionsreduzierung-Methoden für diese Art der Visualisierung ihre eigenen Vorteile haben, konzentrieren wir uns in unserer Software-Implementierung auf Scatterplot Matrizen, um die Daten zu präsentieren. Moderne Informationsvisualisierungstechniken sind der Schlüssel, um eine nahtlose Integration der Scatterplot Ansicht in das Framework zu gewährleisten. Um die Benutzerfreundlichkeit zu maximieren, kombinieren wir mehrere dieser Techniken, um aus einem gewöhnlichen Scatterplot ein leistungsfähiges Werkzeug für die Analyse von multivariaten Daten zu erstellen. Bei der Verwendung von Scatterplots Matrizen können eine beträchtliche Menge Daten anheimfallen. Weil eine schnelle Reaktionszeit in einem Echtzeitsystem eine wichtige Anforderung ist, sind Methoden, welche die Rechenleistungen optimieren, für unsere Implementierung besonders nützlich.

Schlüsselwörter: Streudiagramme, Scatterplot Matrizen, focus plus context, details on demand, Genexpressionsdaten, Caleydo, OpenGL, JOGL

Pledge

I hereby certify that the work presented in this master's thesis is my own and that work performed by others is appropriately cited.

Ich versichere hiermit, diese Arbeit selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und mich auch sonst keiner unerlaubten Hilfsmittel bedient zu haben.

Danksagung

Anfangs möchte ich meinen Eltern, Manfred und Doris Pillhofer danken, die mir meine Ausbildung ermöglichten. Ich möchte mich bei ihnen für die Unterstützung und das Verständnis bei allen meinen Entscheidungen bedanken, und nicht zuletzt für die Förderung meines Wissensdurstes, welcher meine Leben immer wieder bereichert hat.

Ebenfalls danke ich ganz herzlich meinen Betreuern. Dieter Schmalstieg für die Ermöglichung dieser spannenden Diplomarbeit, Marc Streit und Alexander Lex danke ich für das konstruktive erhaltende Feedback, der guten Zusammenarbeit und den wertvollen Tipps.

Ich möchte mich aller herzlichst bei meiner wunderbaren Frau Maureen bedanken, ohne deren Liebe, Unterstützung und Verständnis für die vielen späte Stunden auf der Universität diese Arbeit nicht möglich gewesen wäre.

Zu guter Letzt möchte ich mich bei meinem Sohn Odin bedanken, dessen Lächeln mir immer wieder die nötige Motivation beschert hat.

Contents

Chapter 1

Introduction

If we wish to make a new world we have the material ready. The first one, too, was made out of chaos.

Robert Quillen

When we collect data or gain information in huge quantities, at the first glance everything will appear confusing and chaotic, until we find the means to classify and order it. The field of Information Visualisation provides us with many techniques and methods to detect meanings and patterns in abstract data, find correlations between at the first glance disjointed aspects in the informations we get. But Information Visualisation does not only show us how we reach our goals, a major aspect in this field of science focuses on maximizing efficiency on the way to get there.

Bioinformatics provide a vast playground for Information Visualisation methods. To understand the complexity which lie in biotic systems, computer-aided methods played a big part in the growth of this field the recent years. For example, the analysis of gene expression data, where each experiment can hold thousands of genes and multiple experiments are examined at once to detect important correlations would be a hard, maybe even impossible task without the help of visualisation methods like heatmaps, parallel coordinates or scatterplots.

1.1 Problem Statement and Contribution

At the Graz University of Technology, with the help of the Institute for Pathology of the Medical University of Graz the Caleydo framework was developed, which is an information visualisation framework focusing on pathway exploration and the visualisation of gene expression data.

In the framework there already exist two ways for visualizing gene expression data: **heatmaps** and **parallel coordinates**. It lacks one well-known visualisation method though, which can be typically found in comparable frameworks: the scatterplot. For this reason the focus of this work is to provide a visualisation method for gene expression data based on scatterplots and scatterplot-matrices. In this thesis we describe the methods which we use for a smooth and seamless integration in the framework, the interaction with the already existing methods, the modifications and enhancements we provide over normal scatterplots and the improved usability when using them.

1.2 Biological Background and Gene Expression Data

Like computer programs are encoded in bits and bytes witch describe their behaviour and functionality, the base code on which the living species is built upon is the deoxyribonucleic acid, the **DNA** [Alberts2002]. The DNA, which is a part of the cell, is polymer consisting of 4 monomers (nucleotides) called A (adenine), G (guanine), C (cytosine) and T (thymine). The DNA comes in form of a double helix consisting of a sugar-phosphate backbone, where on each side 2 nucleotides (called base pairs) are connected (A with T, C with G on the opposite strand) [Alberts2002] (see Figure 1.1). How those base pairs are ordered in the DNA represents the building code of a life form, whether it is a small mouse or a human being.

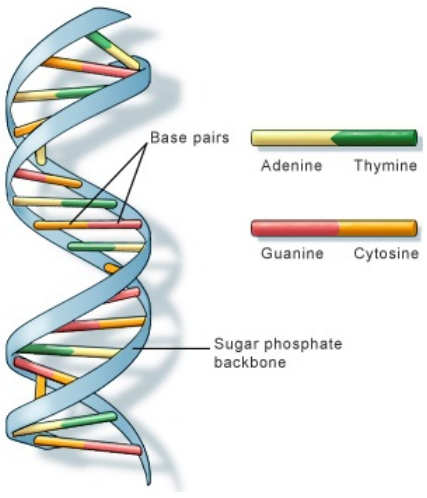


Figure 1.1: *The DNA in form of a double helix and its base pairs (Illustration by the U.S. National Library of Medicine).*

Every cell produces a certain amount of protein. This is called expression regulation. How much protein is produced is encoded in so called non-coding regions. Examining the gene expression regulation can give hints to the functionality or even the malfunction of the studied gene, for example a cancer type can have a distinct regulation pattern [Alberts2002].

To study multiple genes at once **DNA microarrays** became famous in the last 20 years in diagnostics and analysis of the genome. Microarrays (see Figure 1.2) are small glass plates, where thousands of small DNA oligonucleotides called featured are spotted beside each other. After a biochemical process the color of those spots correlate to their expression, which then can be further investigated: detecting differences in the regulations of different patients or of the same patient at different times is a powerful tool for diagnosis and can help with the appropriate treatment of diseases.

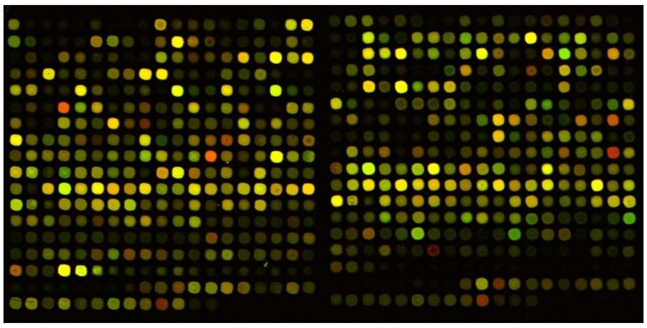


Figure 1.2: *cDNA (complementary DNA) Microarray*

1.3 Structure of this Document

Section 2 will be an overview of related work including general Information Visualization techniques like focus plus context and visualisation of immense data-volumes, scatterplots and scatterplot matrices. Section 3 will show the details about the concept behind the implementation of this thesis and Section 4 will focus on the software design and the actual implementation details. Results and conclusions are discussed in Section 5. Section 6 concludes the thesis and summarizes potential future work.

Chapter 2

Related Work

This thesis is based on two major topics: general information visualisation and in a particular case, visualisation with scatterplots.

Section 2.1 introduces common InfoVis techniques like focus+context, linking and brushing and visualisation of large data-sets. In Section 2.2 we briefly look at the visualisations approaches the Caleydo framework provides, especially focusing on multivariate-data visualisation. Section 2.3 describes the various types of scatterplots, scatterplot-matrices and related techniques like dimension reduction approaches.

2.1 Information Visualization

> ”Visual representations and interaction techniques take advantage of the human eye’s broad bandwidth pathway into the mind to allow users to see, explore, and understand large amounts of information at once. Information visualization focused on the creation of approaches for conveying abstract information in intuitive ways.
>
> *[Thomas2005]*

In simpler words, Information Visualisation (InfoVis) wants to graphically present abstract, often quite complex data as easily to understand as possible for the viewer. One of the first known attempts to fulfil this goal was a planetary movement digram from about 950 A.D. (see Figure 2.1), which shows the changing positions of the sun, moon and the planets in time. Much has changed over the years, from these old diagrams over to Charles Joseph Minard’s *Tableau-graphique* in the 19th century up to modern, sophisticated visualisation techniques, but the common goal always remained the same. However, with computer systems the definition was refined by two aspects: Interaction and animation.

2.1.1 The Information-Seeking Mantra

In his often cited paper, The Eyes Have It:A Task by Data Type Taxonomy for Information Visualizations [Shneiderman1996] Ben Shneiderman introduces us to the information seeking mantra, which even until today did not lose its validity in Information Visualisation. Most applications obey to these rules: *Overview first, zoom and filter, then details-on-demand*. Furthermore, Schneiderman introduces seven tasks users want to perform in an InvoVis environment. Those tasks are:

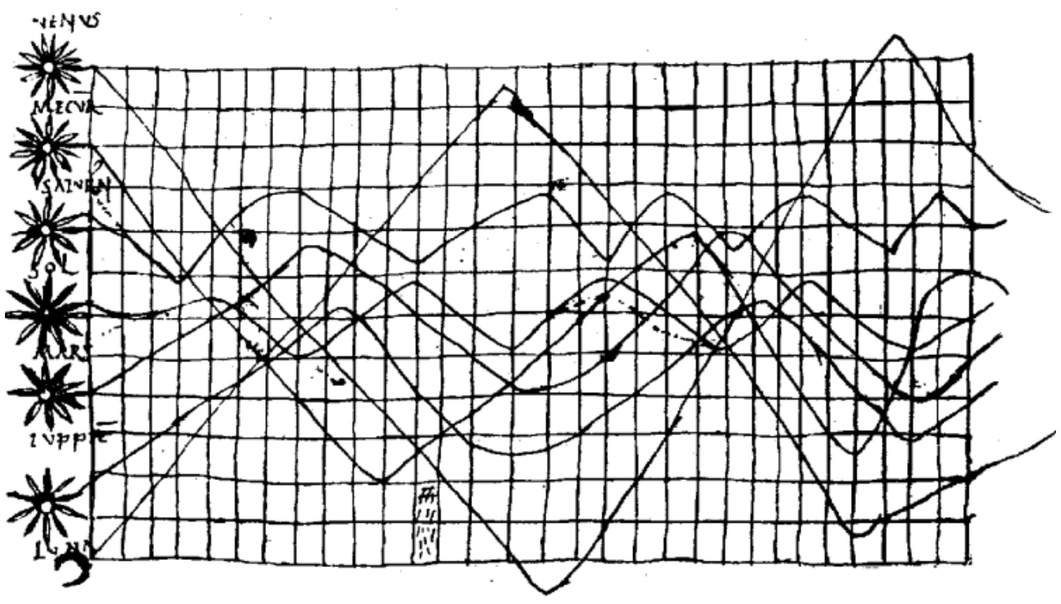


Figure 2.1: *Ancient planetary movements diagram.[Tufte1983]*

Overview

Before you can focus on detail, you need orientation first. A detailed roadmap for example is of no use when you have no idea for what part of the country it is. So you would present the map of the whole country first, before going into detail.

Zoom

After gaining orientation, you want as much detail as possible. Zooming in from the country-map to a simple road map into a detailed street-view let you get your desired information without losing orientation.

Filter

Unnecessary data, which may be useful at other times, can just distract you from finding the particular information you seek. Using the roadmap example, if you wish to find the best way with your car to a certain destination, you don't need elevation-data, you just need to see the roads.

Details-on-demand

Screenspace is valuable, and information should be hidden unless needed to not distract from current tasks. A lot of tools inside a typical graphical user-interface (GUI) are based on this idea: Buttons, mouse-overs, pop-ups, menus and so on provide additional information through user interaction which are hidden otherwise.

Relate

To further help with orientation, emphasizing relationships between the different data sets helps. Multiple coordinated views of different data or different representations shown together of the same data will produce an easier understanding of correlation in the data. Multiple coordinated views are discussed in Section 2.1.3 When highlighting a point of interest (POI) in one view also highlights the same POI in the other views, we talk about *Section Linking and Brushing*, which we discuss more thoroughly in Section 3.4.

History

In lots of interactive computer applications, especially in information visualisation, the desired end-result is rarely accomplished by following a predefined plan, it is often the result of trial and error methods and successive refinements through user actions. Such applications require an undo-functionality for usability-purposes, so having a history of recent events to jump back is needed.

Extract

Having found the desired information, extracting it is often useful for further and deeper investigation or to share them with others. So the ability of saving/printing/drag and dropping sub-collections of the main-data is a commonly used features [Shneiderman1996].

2.1.2 Focus plus Context (F+C)

Schneiderman mantra does not necessarily need to be interpreted as a sequential approach: First we get an overview of the data, and only afterwards we focus on the detail. We can also do this simultaneously. The idea is, to show some parts of the screen in greater detail (Focus) while still keeping a global orientation view with reduced detail (Context), showing all information available. The information in the context-region may also be different from the information wanted in the focus part [Card1999]. Robert Kosara and Helwig Hauser describe four groups of F+C techniques[Kosara2003]:

- Distortion-oriented.
- Overview Methods.
- Filtering.
- In-Place Techniques.

Distortion-oriented F+C methods provide more space for the detail view simply by geometrically distorting the whole or just parts of the image [Kosara2003]. They are discussed in more detail in Section 2.1.2.1.

In **Overview Methods**, Focus and Context are presented separately in different regions or in different windows. The default-setup of the windows-explorer is a good and well known example here, its shows the whole directory-tree of the file system in one part of a window, and the contents of one directory with more details on the other half.

With **Filtering** methods, with the help of user interactions, additional information is filtered out of the overview. Magic Lenses [Bier1993] provide such an approach by letting the user move a special shape over the window which reveals additional details (see Figure 2.2). Another method was presented by Kosara et al.[Kosara2002]: To emphasize relevant information to the user, they introduced a method that blurs object based on their relevance which they called *Semantic Depth of Field (SDOF)*, inspired by the depth of field (DOF) effect used in photography (see Figure 2.3).

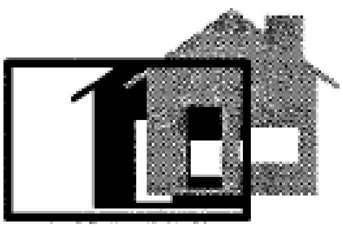


Figure 2.2: *The rectangular 'Magic Lense' dragged over this house-sketch reveals its shadows. [Bier1993]*

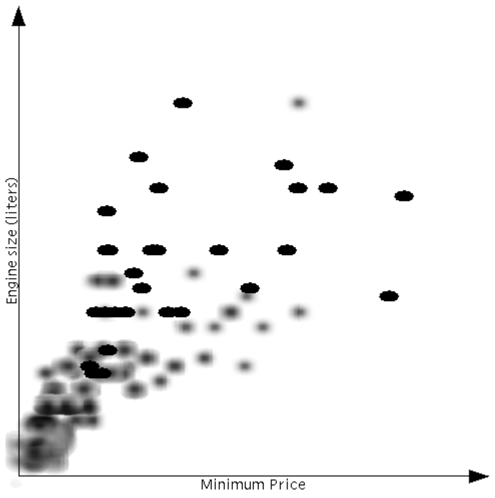


Figure 2.3: *This simple scatterplot shows the Semantic Depth of Field (SDOF) in action. The at the moment not relevant information is blurred, so it can be better distinguished from the desired details. [Kosara2002]*

In-Place Techniques do not require a lens, they hide and show different informations in the same view depending on user interaction. Some Level-of-Detail (LOD) techniques are also part of this group. (LOD means that each piece of information can be displayed in various degrees of detail, using the level which is mostly appropriate in the given situation).

2.1.2.1 Distortion-oriented F+C Methods

Since one of the foci of this thesis practical part employs some of these F+C techniques, we introduce them here in more detail.

Fisheye Views

The most popular methods are the fisheye views [Furnas1986], derived from the fisheye lens used in photography. The fisheye-view shows the (in most cases interactive) focus area larger than the rest of the image, using distortion to avoid occlusion or obscuration. Major drawbacks of fisheye views are that distortion also affects the focus-area, which makes some visualisations difficult to interpret. An example for a fisheye-distorted graph can be seen in Figure 2.4.

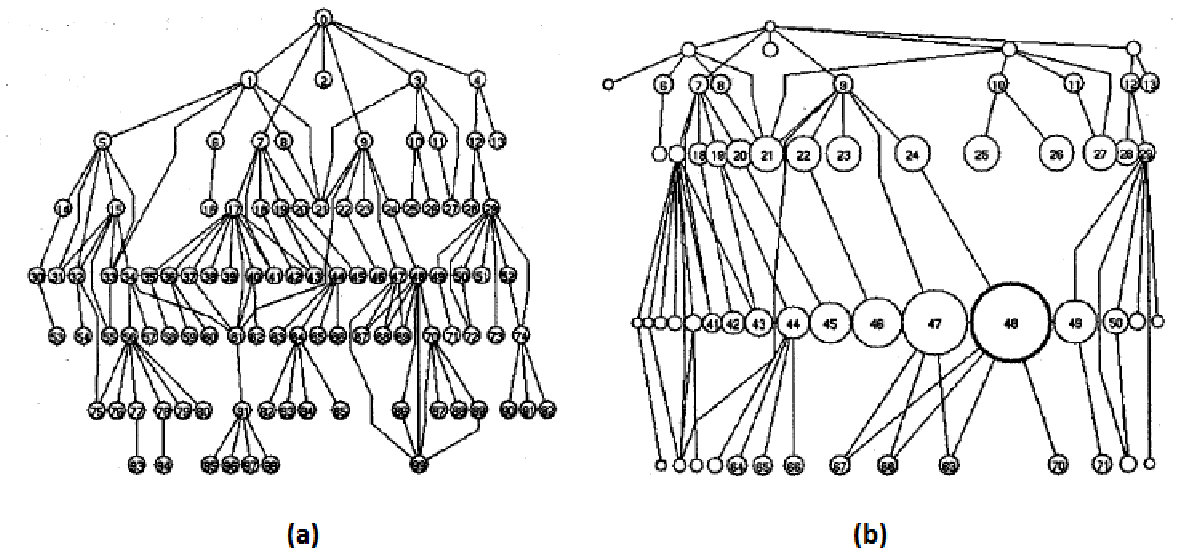


Figure 2.4: *A graph with 100 vertices and 124 edges (a), (b) shows the fisheye-distorted graph with the focus on the '48' label.[Sarkar1994]*

The Perspective Wall

The perspective wall is a visualisation method developed by Mackinlay et al. for viewing mostly 1-dimensional data (e.g. Directory entries)[Mackinlay1991]. The method maps a 2D-Layout onto a three-parted 3D-Wall, resulting in 3 panels, the middle, nearest wall as non-distorted focus-panel, the other two are distorted into perspective (see Figure 2.5). Because there are just 3 flat panels, this is an easy to implement approach with standard hardware using simple texture mapping [Kosara2003].

The Document Lens

The document lens [Robertson1993] follows a very similar approach. The idea is to view an entire multiple-page document on one screen while still being able to read. A normal magnifier lens would obscure regions of text, while using a fisheye lens will distort the text and make it harder to read. The document lens avoids this: Here, the rectangular magnifying lens represents the focus area, where the region outside the lens is distorted into the perspective away from the screen (see Figure 2.5). The view looks like a truncated pyramid from above, allowing the user to navigate through all the pages.

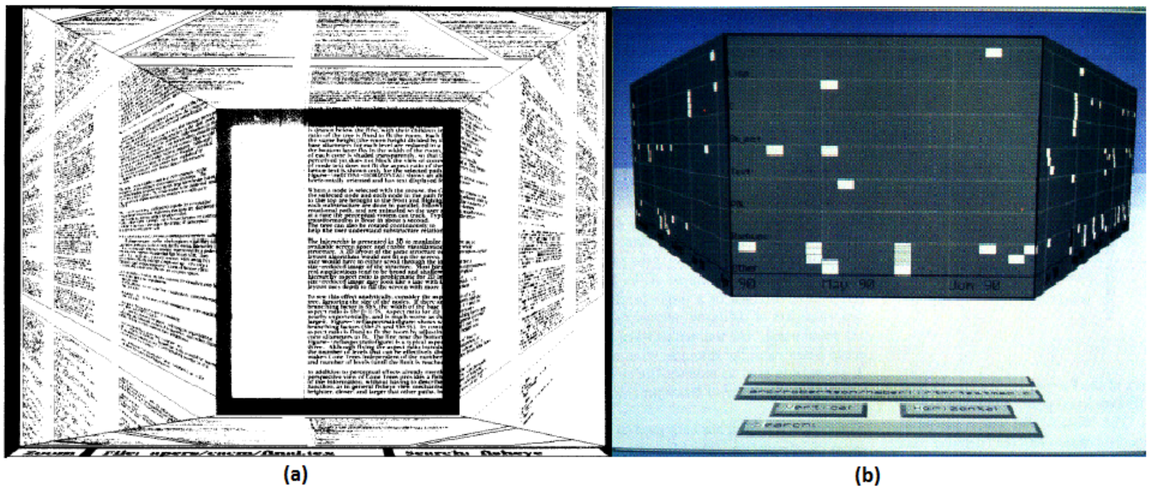


Figure 2.5: *(a) The Document Lens and (b) the Perspective Wall. Both techniques show the focus-part as non-distorted area.[Mackinlay1991][Robertson1993]*

Orthogonal Stretching

In the orthogonal stretching method [Sarkar1993] the image is divided into rectangular subspaces, forming a grid. The grid lines are called *(bar) handles*, which are used to stretch the screen from the handles point. Moving such a bar handle, the screen is stretched on one side and expands on the other side. With this system, the user is able to gain multiple focus areas on the screen, while still obtaining the orthogonal ordering of the regions and keeps symmetry (see Figure 2.6).

A related technique is the polygonal stretching, which allows arbitrary polygonal focus regions, but forfeits the orthogonal ordering [Sarkar1993].

2.1.3 Coordinated Multiple Views

Multiple Views are commonly used to present the same data under different aspects in two or more windows. Baldonado et al. proposed guidelines when and how multiple views should be preferred to single views [Baldonado2000]:

- Diversity: When different views show diverging aspects of the data.
- Complementarity: The different views emphasize correlation or disparities of the data.
- Decomposition: If a view gets too complex for the viewer, separate the data in different views.
- Parsimony: When a new view does not justify the cognitive context switching latency, the space and/or the computational costs which multiple view raise over single ones, one should not use them.
- Self-Evidence: Show the relationship between different views.
- Consistency: The user interface and the state of the data in different views should

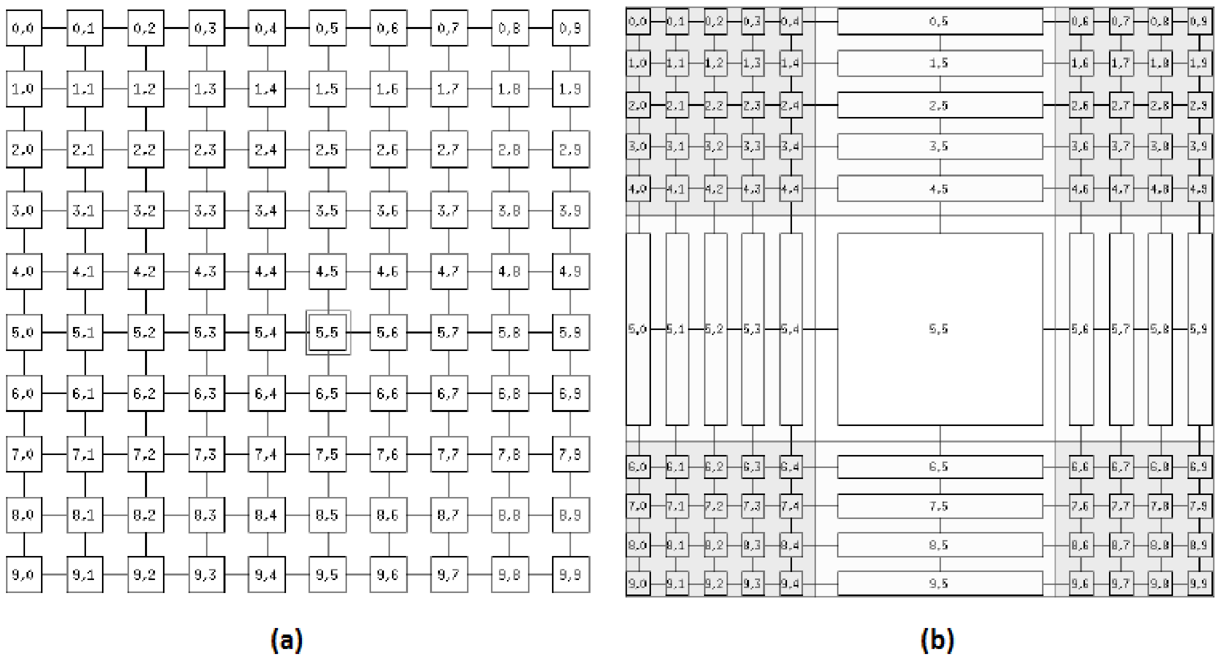


Figure 2.6: *(a) shows the original visualisation, focus on the square labeled with '5.5'. (b) demonstrates F+C with orthogonal stretching, using four bar handles .[Sarkar1993]*

be consistent

- Attention Management: The focus of the user should be directed to right view.

Multiple views are well-suited for overview-methods [Roberts2007a] as discussed in 2.3.5. Plumlee and Ware showed, by comparing user perceptions between zooming and multiple window interfaces, that the number of user errors by using multiple window interfaces have been much lower than with zooming interfaces [Plumlee2006]. The reason they gave was that users utilize different windows more often than using a zoom feature for comparison tasks. They also concluded that using zooming interfaces is faster for the test subjects with few objects to compare, but with more objects a multiple windows environment is faster.

Limiting multiple views to two instances, we speak of **dual view** systems [Convertino2003]. We can distinguish here between the above mentioned **focus+context views**, the similar **overview+detail** techniques where the overview windows shows more simplified and abstracted data [Roberts2007a]. Further we have **difference views** which emphasize on the varieties in data. **Master/slave** systems let one view navigate others. **Small multipes** are small views layed side-by-side spawning a matrix. They use the same visualisation techniques but show different subsets or different parametrized data. This helps to identify the influence one parameter has on the data.

Coordinated multiple views are not only build on the same base-data-set of which they derive their views, but the different visualisation are also concatenated, logically and often visually. Coordinated multiple views should not be seen isolated, they should be treated as an entity so that *"..information contained in individual views can be integrated into a*

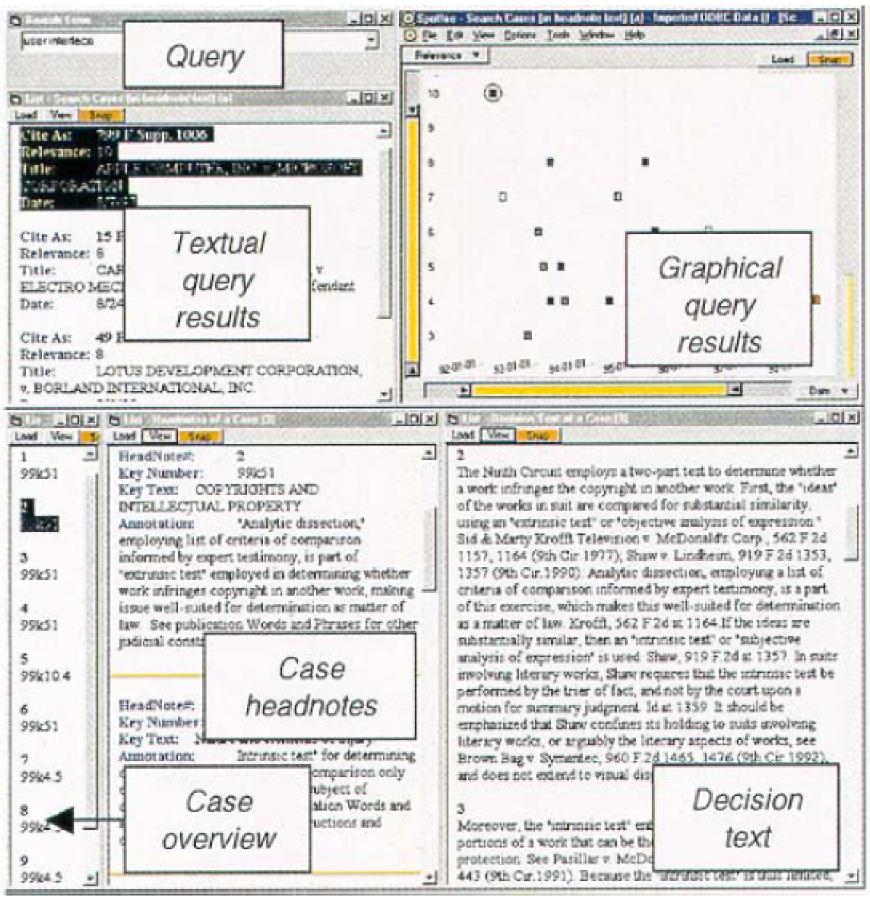


Figure 2.7: *Multiple Views.[Baldonado2000]*

coherent image of the data as a whole" [Buja1991]. To visually emphasize the correlation of each view, several methods exist: Navigational slaving is one technique, where movements in one view are propagated to the others [Baldonado2000], Visual links [Shneiderman2006, Aris2007, Collins2007], which are drawn lines between correlated data in different views, is another technique. Their use in the Caleyedo framework is discussed in [Lex2008]. The last approach, *Painting multiple views* [Buja1991] or better known as *Linking and Brushing* is described in the following chapter.

2.1.4 Linking and Brushing

The idea of linking and brushing is to combine different visualization methods to overcome the shortcomings of single techniques. Interactive changes made in one visualization are automatically reflected in the other visualizations. Note that connecting multiple visualizations through interactive linking and brushing provides more information than considering the component visualizations independently.

[Keim2002]

Selecting a sub-set of the visible data-items with a mouse or similar tools is called brushing. The data-subsets are typically highlighted (or color coded) to distinguish them from the rest of the data. If those sub-set are also highlighted in other views, the brushes are considered linked. For example using a rectangle-selection in one plot of a scatterplot matrix shows the corresponding points in the other cells as well [Becker1987] (see Figure 2.8).

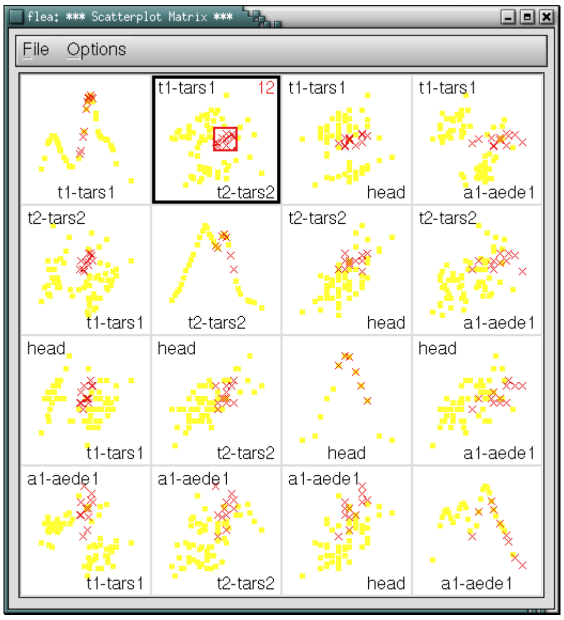


Figure 2.8: *Linking and Brushing in a scatterplot matrix, using a rectangular brush.[Voigt2002]*

The basic brushing tools for 2-D environments consist of the same tools found in graphics editing programs. Various shapes are drawn with a mouse around the desired selections, examples are rectangle shapes, lasso, freeform shapes, rubber-bands, bezier curves, polylines and so on. More special presentation may also provide adjusted brushes, like the *angular brushes* in parallel coordinates [Hauser2002] or special 3D-brushes for 3-dimensional visualisations.

On the example of XmdvTool [Ward1994], Martin et al. dig deeper in the details with brushing technique [Martin1995]. XmdvTool allows to manipulated the brushes after setting them (e.g. a brush can be hovered across the screen by dragging it with a mouse), multiple brushes and their interactions with boolean operations are featured and new brush operations are introduces like averaging or masking [Martin1995]. They further distinguish between *demand driven brushing* (like explained above) and *data driven brushing*, where the sub-selections are not decided by user interaction only but also by the data lying behind, which helps to form the brush. For example, in a heavy cluttered area, it may be hard to decide which data should lie outside or inside a brush. XmdvTool helps the user by introducing a virtual paintbrush with witch the user can paint on the data-points of interest. The painted points then form the brush.

Another issue arises when you don't want the decision if a point is in a selection or not to be binary [Martin1995]. The XmdvTool solves this by using brush boundaries, where each point inside the brush gets a degree of interest (DOI) value depending on the spatial distance to the boundaries. The brushed data is then visualized depending on its DOI value. In the paper by H.Doleisch and H.Hauser [Doleisch2002], they built on this idea. They called their method *smooth brushing*, inspired by the smooth data gained from flow simulations. In volume rendering, to deal with occlusion problems in 3D-Space, the rendered data points are often given an opacity value provided by a transfer function. The DOI function corresponds to this opacity value.

Hong Chen further generalized the brushing strategy with his *compound brushing* [Chen2004] idea, which consist of five components (data, selection, device, renderer and transformation) to model most kinds of brushes for different visualisations.

2.1.5 Efficient Rendering of huge data-sets

Although Moore's Law [Moore1965] is still in place and will be at least for the next 5-10 years, the sometimes huge amount of data, especially with high-dimensional data-sets, raising complexity in visualisations and coordinated multiple view strategies still demand other strategies then naive brute-forcing visualisations [Andrienko2007].

In 2005, Keim et al. proposed an extension to Shneiderman's information-seeking mantra to "Analyse First - Show the Important - Zoom, Filter and Analyse Further -Details on Demand", which become known as *Visual Analytics Mantra* [Keim2005]. Seen in this context, this can also be interpreted as if there are already many analytic steps early in the visualisation process, the actual visualized data load can be already greatly reduced to only show the important information. In other words, the analytic steps could also include performance considerations to ease scaling issues.

Still, often there is the desire to view a huge, unfiltered data-set at once, while maintaining high responsiveness (below 100 ms [Shneiderman1994]) for user interactions in dynamic queries.

Sampling

One possibility to reduce data-load is statistical sampling of the data. Information may be lost, but for some cases this can vastly improve computational and often even visual performance [Dix2002]. Statistical sampling should be used when there are too many points to fit on the screen, details are not really that important (overviews) or if information is lost anyway due to filters like averaging [Dix2002].

Layered Visualisation

Another method to reduce the actual rendering time, is *Layered Visualisation* described by Piringer et al [Piringer2009]. The final image is divided into separate elements, spatial and also chronological in the visualisation pipeline. These subdivisions are called layers and should be able to be rendered as independently of each other as possible. To be more precise, they distinguish between:

- Semantic Layers: Spatial and logical different parts of an image
- Incremental Layers: divide the data into sampled subsets where all layers together

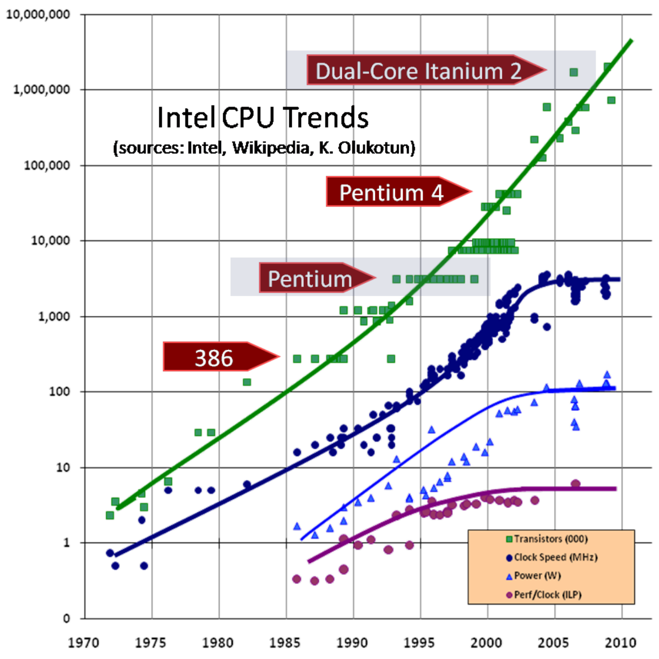


Figure 2.9: *Intel CPU Chart. Notice how clock speed did not rise as fast as the transistor count due to physical limitations. This is the main reason why there was a great paradigm-shift to parallel computing [Sutter2005].*

present the final image

- LoD Layers: lower LoD-levels with less visualisation costs are replaced by higher LoD-layers when time permits it.

The advantage this technique provides is that by each user-interaction or changes in the base dataset, often only a small portion of the final image needs to be updated, so a lot of the layers can be reused without costly re-calculations or re-renderings.

Multi Threading and Early Thread Termination

Even using the methods mentioned above, we are still limited by what the CPU and the graphics hardware allow. Nevertheless, out of necessity [Sutter2005] (See Figure 2.9) and the success of parallel graphic hardware like NVIDIAs CUDA [1] (Compute Unified Device Architecture), the nowadays very popular parallel software design is going to influence InfoVis. An obvious example is the rendering pipeline, which is now almost fully shifted into hardware, and since each subset of the image can be rendered often completely independent of each other, it can be very effectively parallelized.

All of the methods described have still one big disadvantage: for seamless visual feedback the final anticipated image quality has to be determined beforehand for each task depending on computational power. Since the amount of input data can change during the application process, the final goal has to be rather conservative or it risks response time issues. To solve this, Piringer et al. combined the techniques of layered rendering with a multi-

[1]http://www.nvidia.com/object/cuda_home_new.html

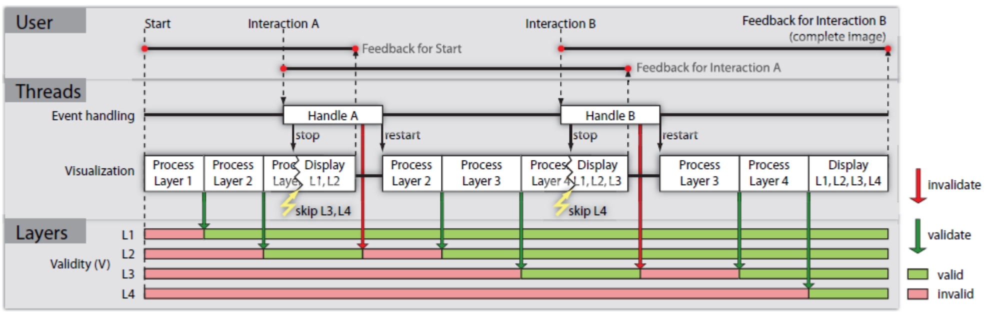


Figure 2.10: *Early Thread Termination and Layered Visualisation [Piringer2009].*

thread approach [Piringer2009]. Using the *Active Object* design pattern [Schmidt2000], they let each layer be rendered when triggered by an event, and stopped or restarted the visualisation process if the event occurs again and the layer has not finished yet (see Figure 2.10). They made sure to chose layers as efficiently as possible to be able to show partial results during constant user interaction (e.g. moving a slider), which could be problematic otherwise if a rendering task would have to be restarted constantly, thus never showing any results.

2.1.6 Discussion

For displaying huge amounts of data, especially multivariate data, early design decisions are inevitable. Thankfully, the field of InfoVis equips us with a many tools and techniques to simplify this task.

Building up on Shneiderman mantra [Shneiderman1996], we can dissect our intent into smaller, manageable parts. Focus+Context methods let us split our attention on the on overview and detail visualisations, which often require completely different design decisions. Overviews should provide an overview, a feel of the scale and they should serve as orientation guide for the actual data, occlusion or even loss of data is not that important here. On the other hand, visualizing the details in the data should guarantee that the user has access to any bit of information which is available and that data-points are distinguishable from each other.

Linking and Brushing help us overcome the interactive challenges which arise while using multiple views. Finally, techniques as discussed in the last section let us overcome the constrains which hardware-limitations and vast input-data produce.

2.2 InfoVis in Caleydo

Caleydo[2] is an information visualisation framework focusing on pathway exploration and the visualisation of gene expression data. Its based on Java[3] and Java Bindings for OpenGL (JOGL)[4]. It uses many advanced InfoVis techniques which where already discussed in Section 2.1:

- Multiple Views
- F+C, especially worth mentioning the Bucket, described in [Streit2009a] and [Lex2008].
- Linking and Brushing
- Details on Demand
- Visual Links

Pathways, which describe biological processes in cells, are not in the scope of this work and are thoroughly discussed in [Streit2007] and [Streit2008], the interaction with gene expression data is the topic of the [Lex2010] paper. Caleydo provides many visualisation modes like **Histograms**, **Dendograms**, **Radial Hierarchy Layouts** and others, all implemented as separated views for a customizable application thanks to a plug-in architecture. Since the practical part of this thesis focuses on the visualisation of gene expression data with scatterplots in this work, we briefly describe the two other visualisation methods that currently fulfil this task:

2.2.1 Heatmaps

Heatmaps are the most common visualisation methods for gene expression data, developed 1998 by Eisen et al. [Eisen1998]. In a heat map, each data-point is visualised as a small, color-coded (mostly green, black and red, but other colour schemes exist) square, depending on its value. Their strength is their easy reordering of its cells according to clustering algorithms, and they are therefore often drawn together with dendograms, a tree-structure visualisation. Caleydo uses an enhancement by Schlegl with an hierarchical approach [Schlegl2009] (see Figure 2.11).

2.2.2 Parallel Coordinates

Parallelcoordinates (PCs) are a popular technique for the visualisation of multivariate data, invented by Maurice d'Ocagne in 1885 and much later further developed by Inselberg [Inselberg1985]. PCs map each data row on parallel axis, and each point on the axis is connected with its corresponding points on the adjacent axes. This means each n-dimensional data point is visualized by a polyline across all n axes. The PCs in Caleydo (see Figure 2.12) support angular brushing, let the user modify the axis and handle cluttering issues with the help of transparency and random sampling [Lex2008].

[2] http://www.caleydo.org/
[3] http://java.sun.com/
[4] https://jogl.dev.java.net/

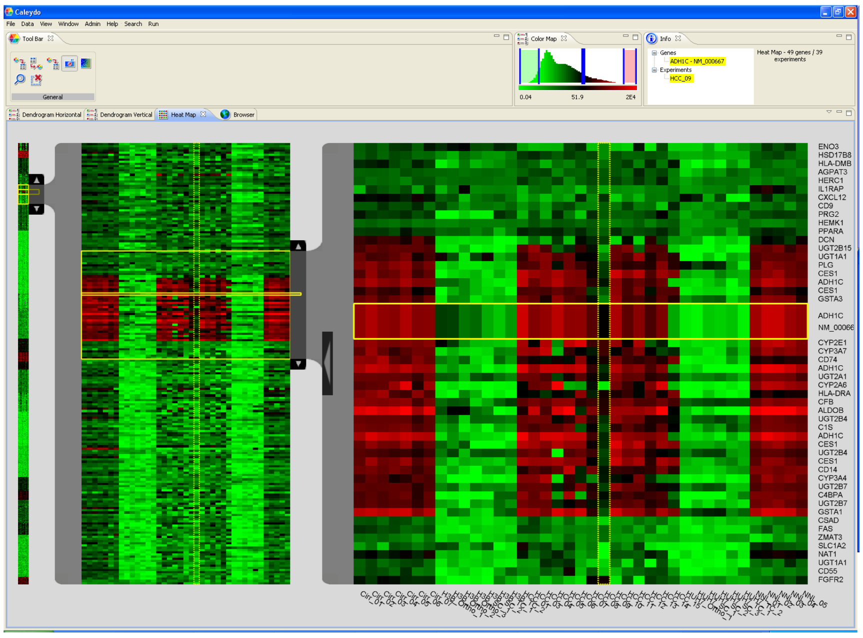


Figure 2.11: *Hierarchical Heat Map in Caleydo*

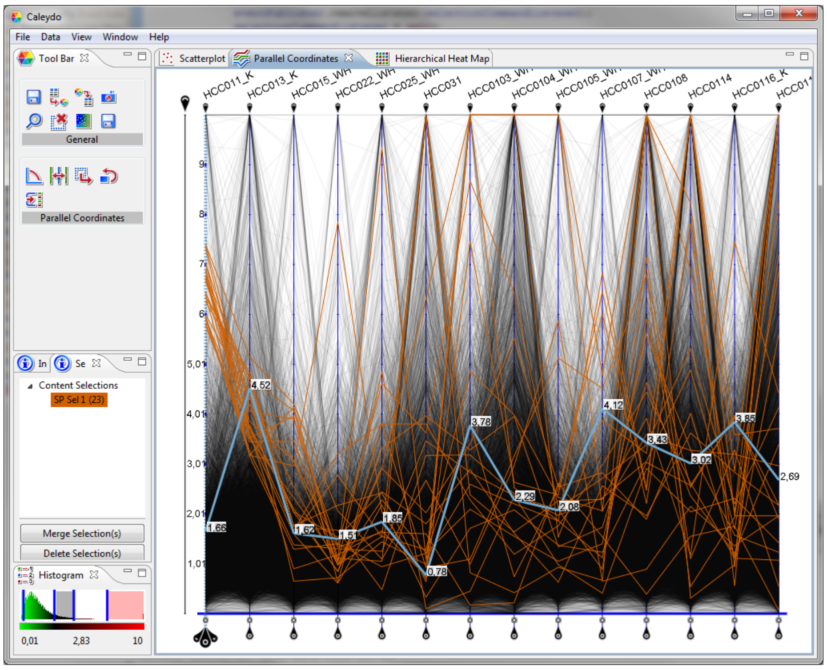


Figure 2.12: *Parallel Coordinates in Caleydo*

2.3 Scatterplots

Scatterplots are a widely known, simple and easy to understand form of a diagrams used in statistics and information visualisation. Scatterplots, also known as scatter diagrams or X–Y graphs, are also commonly used in quality control, described as one of the seven ”indispensable” tools [Tague2004].

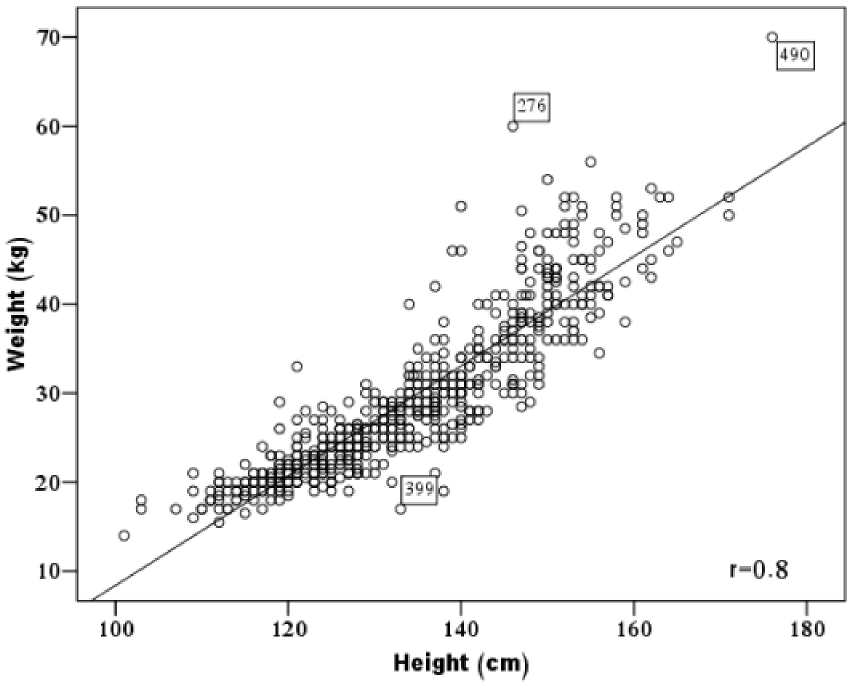


Figure 2.13: *A scatterplot showing a linear positive height-weight correlation of 606 Afghani pupils aged 6-14 years. Outliers are labeled with their case number.[Rezaeian2009]*

A scatterplot is a diagram, where each of the 2 dimensional data points is mapped on the cartesian coordinate-system. In other words, the value of the data point represents the position in the diagram. The value is then visualized by a point or glyph. Scatterplots are used to indicate different kinds of correlations between variables. The further individual points are away from the diagonal, the greater is their variance.

If there is a correlation, clustered points will indicate a line, which is called trendline or line of best fit. If the line is linear, the trendline is also called linear regression curve. If the slope of the curve is raising, we talk about a positive correlation, if it is falling, we have a negative correlation. Scatterplots do not have to form trendlines to reveal meaningful results, a cluster indicates already by itself that the variables in this areas are showing a strong similarity.

One of the interesting aspects of scatter plots is the fact that you can also see nonlinear correlation trends at the first glance, and if the plot is randomly cluttered, we see immediately that there may not exist any significant correlations. Another advantage of this visualisation mode is that you can easily spot points not fitting the trendlines, called outliers.

An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs. [Grubbs1969]

2.3.1 Overplotting

While large datasets obviously put constrains on the speed, complexity and display quality [Murrell2006], one problem especially related to scatterplots is overplotting. With a dense data-set, each point may lie so close to each other that one can't distinguish if there are just some or a huge amount of points in a specific local area.

Reducing the point size may help in certain cases, but for plots where the amount of data-values exceeds the diagram-resolution greatly in some highly cluttered areas, this methods won't really help. Another disadvantage of reducing point size is that while the cluttered area may show some meaningful information, outliers are hard to see or may even vanish. This is also true when using transparency values for each point or glyph, an otherwise simple but very powerful technique to show the density in plots.

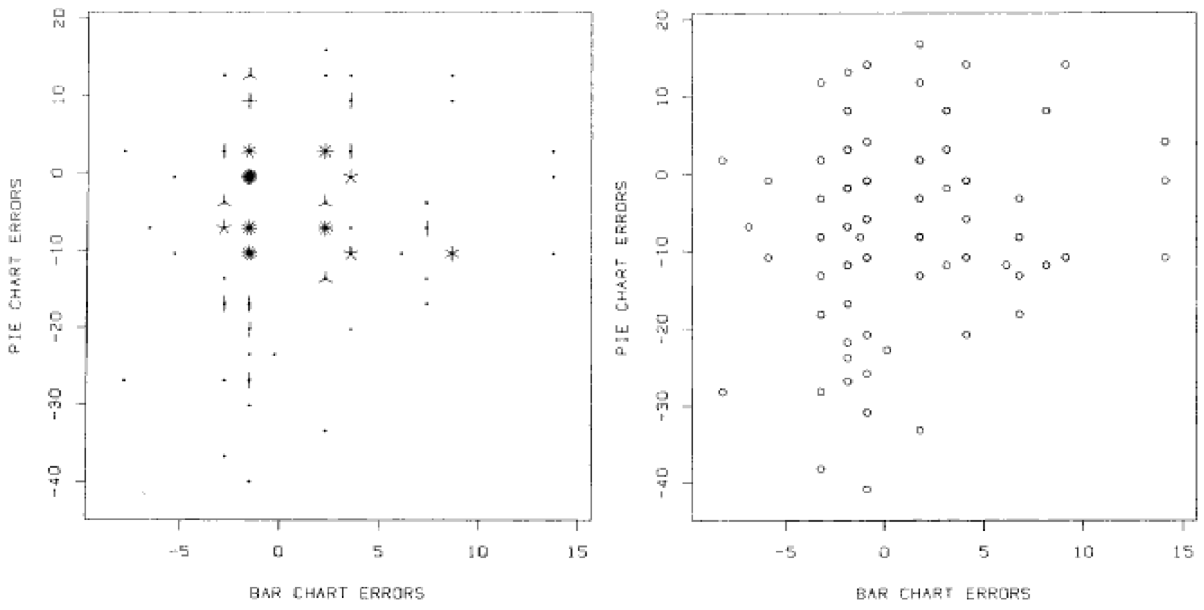


Figure 2.14: *Left:sunflower plot; right: scatterplot. Due to overplotting, only a fraction of the points could be seen in the original scatterplot.[Cleveland1984]*

To solve this problem, we can span a virtual grid over the scatterplots. Now we count the points in each grid-cell and fill the cell with a color, symbol or transparency value related to the point quantity. This is called binning. One example of this plot is the sunflower plot of Cleveland and McGill [Cleveland1984]. There, the grid-cells are represented by symbols called 'sunflowers', which consist of line-segments (called petals) radiating around the center of the grid-cell. The number of pedals in each sunflower corresponds to the number of points the related cell. (See Figure 2.14).

The drawback of sunflower-plots is, that specially with bigger cell-sizes, the exact location of the points in low-density regions are lost [Dupont2003]. To solve this, one can draw not only sunflowers but also the exact points in locations, where there are only a few points in a grid cell [Carr1987]. To further improve this type of visualisation, one can also use a hexagonal tiled grid to squeeze more bins into the plot, and to make strictly horizontal and vertical patterns not as visible [Dupont2003]. Using color coded shapes result in *Density Distribution Sunflower Plots*, as suggested in [Dupont2003] (see Figure 2.15).

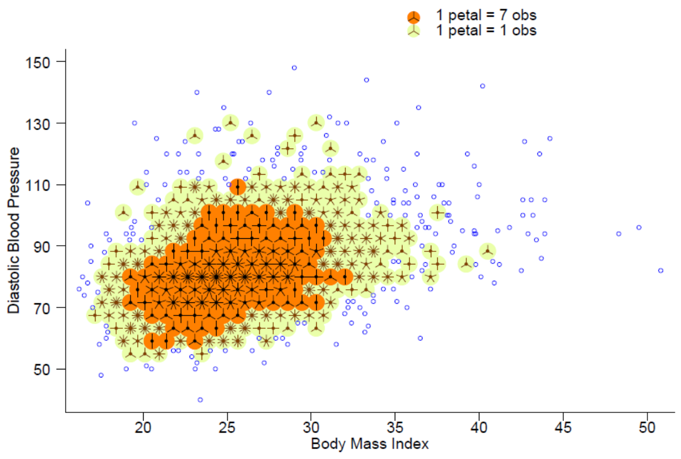


Figure 2.15: *Density Distribution Sunflower Plot. The diagram consists of regular hexagonal bins consisting of color-coded sunflowers. Blue circles are drawn at their exact location as long as there are less than 3 points per hexagon [Dupont2003]*

2.3.2 Scatterplots related diagrams

Normal scatterplots can visualize only two-dimensional data. To raise the dimensionality by one, there is the possibility to connect each point with a line by the increasing order of a third variable [Rabenhorst1994], leading to *parametric snake plots*, which looks like a connect-the-dots picture. The deviation from a straight line "indicates variability introduced by a third variable on the relationship between the other two" [Schall1995] (see Figure 2.16).

Similar to the snake plot is the quad-vise plot, where two scatterplots are drawn beneath each other and each point in one scatterplot is connected with a line to the corresponding point in the other scatterplot. Thus, the line shows the relationship between 2 pairs of variables, so effectively showing 4 dimensions [Schall1995] (see Figure 2.17).

2.3.3 Continuous Scatterplots

One limitation of scatterplots is that they are only useful for discrete data-input. Sven Bachthaler and Daniel Weiskopf enhanced this model by introducing *Continuous Scatterplots*, which enhance the diagram to provide a visualisation of spatially continuous input data [Bachthaler2008]. But they are also useful for displaying discrete data, for example

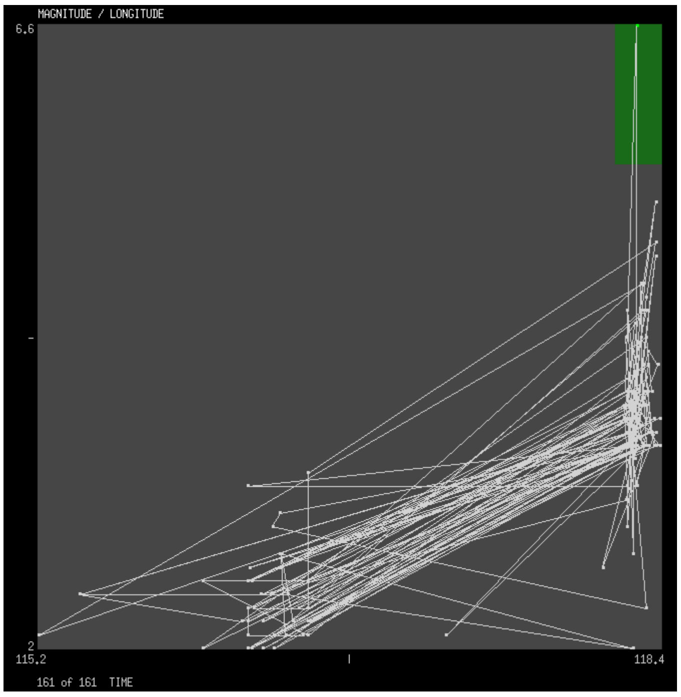


Figure 2.16: *Parametric Snake Plot in DIAMOND, a tool for interactive exploration of multidimensional data.[Rabenhorst1994] The line indicates the influence of the third variable.[Schall1995]*

they do not have the problems of overplotting. Because continuous scatterplots are based on scalar density functions, scalar field visualisations like isosurfaces can be applied here [Bachthaler2008] (see Figure 2.18).

Relate: Isosurfaces and Contour plots

An isosurface is a surface representing scalars in a 3-dimensional scalar-field. While isosurfaces (and their 2D-pedants, *Contour plots*) are more known in scientific visualisation and 3D-reconstructions of point clouds or computational fluid dynamics (CFD), they are also known to be used in InfoVis. For example, the work of Bajaj et al.[Bajaj1997] describes an interactive approach in isosurface statistics, while Carr et al.[Carr2006] investigated the influence of the higher interpolation methods (which isosurface statistics provide), to histograms, which use nearest neighbour interpolation. Examples are illustrated in Figure 2.19.

2.3.4 Multivariate Data and Scatterplots

3D Scatterplot

In the chapter before we focused on 2-D scatterplots, but of course it is an easy task to extend the idea by adding another coordinate axis, leading to a 3D-scatterplot (see Figure 2.20). Expanding the plot into the third dimension will yield to some other problems, though. Overplotting issues are emphasized heavily by occlusion; user interaction to

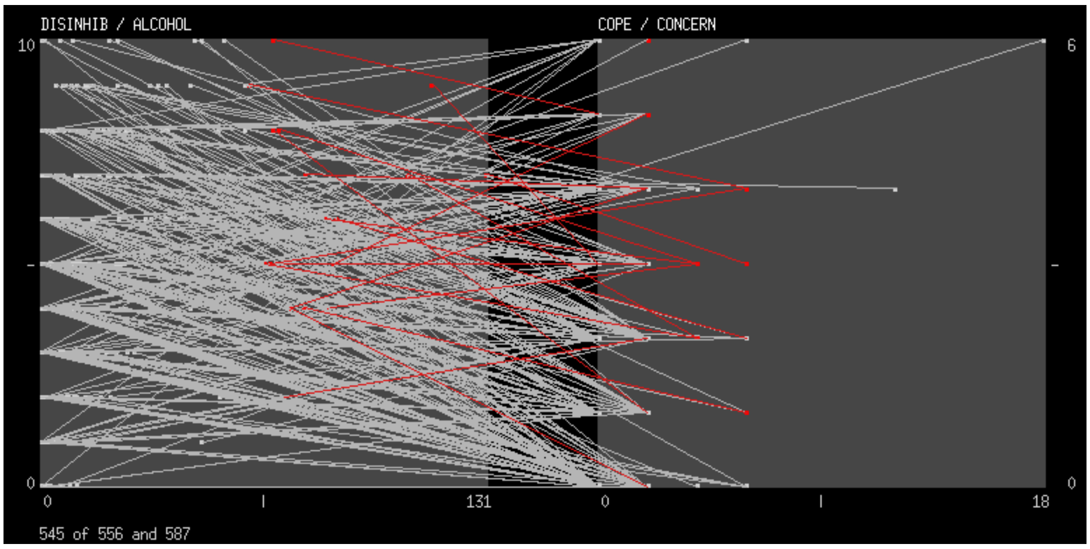


Figure 2.17: *4-dimensional linked quad-wise plots in DIAMOND, a tool for interactive exploration of multidimensional data.[Rabenhorst1994] The lines between the two plots shows the correlation between 2 pairs of variables.[Schall1995]*

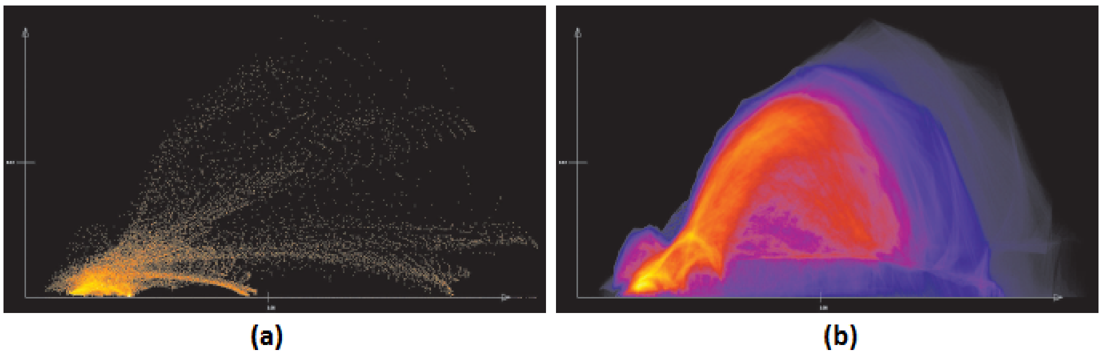


Figure 2.18: *(a) shows a the discrete scatterplot, where (b) shows the continuous version of the same data-set. [Bachthaler2008]*

find the desired view-angle is more or less required and interactions with sub-sets require more sophisticated tools like a simple rectangle-brush. For this reasons, "..their perceptual effectiveness is unclear" [Bachthaler2008]. To alleviate these problems a little, the main 2D-projection of the 3D-plot in -x,-y and -z direction are sometimes additional visualized for better orientation.

As discussed in the last section, scatterplots provide a fine solution for 2 dimensions of the input data (bivariate data). With enhancements like 3D-scatterplots, snake-plots and quad-vise plots we can raise the dimensionality a little. But for multivariate data-sets like gene expression data for many experiments, scatterplots are very limited. Nevertheless, there are methods to squeeze additional dimensions into one scatterplot, some methods are suggested in [Cleveland1988]:

- use different sizes for an additional dimension

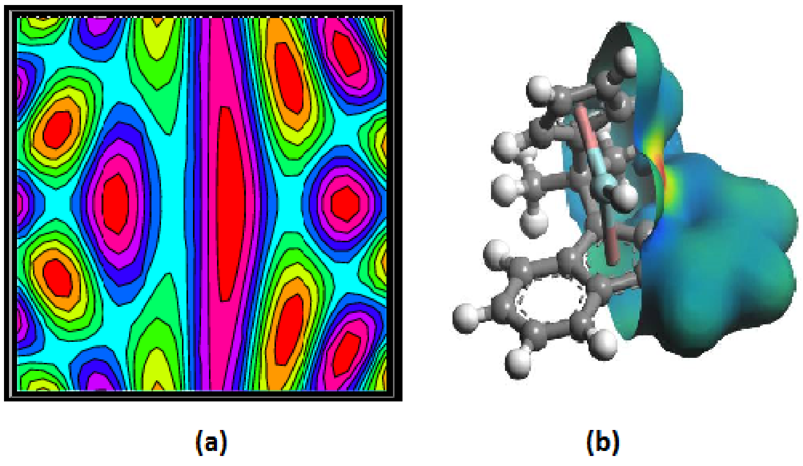


Figure 2.19: *(a) Weisstein, Eric W. "Contour Plot." From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/ContourPlot.html (b) shows an iso-surface around a zirconocene molecule. (Accelrys (http://www.accelrys.com))*

- use different shapes for an additional dimension
- use different spatial orientation of shapes for an additional dimension
- use different colours for an additional dimension

In other words, any *Visual Variable* can be used here: Jaques Bertin [Bertin1974] described methods to modify basic units called marks. Those methods are called Visual Variables and methods are shapes, sizes, textures, values, colors, orientation and position.

Unfortunately, all of these methods have their disadvantages. Too much shapes/colours make the plots quite unclear and confusing. It is also quite important which shapes are used, as they allow a wide range of discriminability [Li2009]. Furthermore, techniques discussed previously circumventing the overplotting problem cannot be applied that easily. Shapes or colours may be also already reserved for sub-selections (brushing), clusters or other things. While there is the possibility to squeeze up to seven dimensions into one plot [Schall1995], in practice 2-dimensional data will be the limit for most common scatterplots.

Nevertheless, other solutions for showing higher dimensional multivariate data with scatterplots exist:

- Dimension Reduction Methods: Building a linear combination of the whole or part of the dataset and mapping them onto 1 scatterplot, for example with the Principal Component Analysis.
- The Scatterplot Matrix: Mapping each axis of the n-dimensional dataset on each other to gain a nxn-Matrix of scatterplots.
- A combination of dimension reduction and still having multiple different plots of the same base-data, ordered by their entropy.

We will discuss these methods in the following sections.

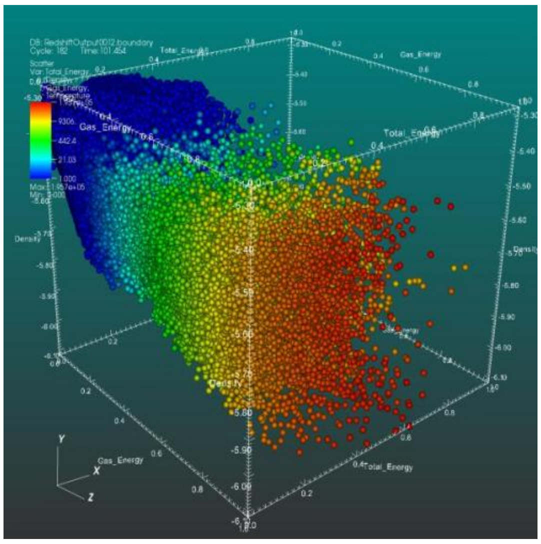


Figure 2.20: *3D-Scatterplot 4th axis mapped into one 3D Scatterplot. The color-coding represents the fourth dimension.*

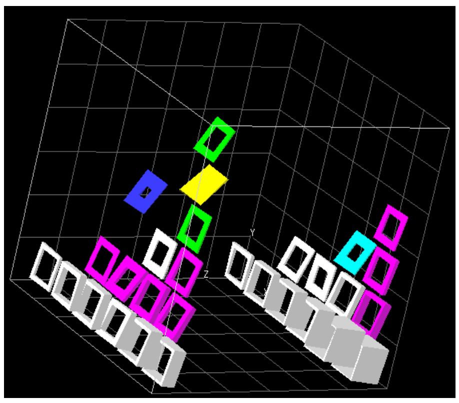


Figure 2.21: *3D-Scatterplot showing seven dimensions.[Schall1995]*

2.3.4.1 Dimension Reduction Methods

Principal Component Analysis

The Principal Component Analysis was introduced in the year 1901 by Karl Pearson [Pearson1901] and further developed by Harold Hotelling 30 years later. It is also called *Hotelling transform*, *discrete Karhunen–Loève transform (K.L.T.)*, or *proper orthogonal decomposition (POD)*. The PCA is an orthogonal linear transformation to maximize the variance of the original data-set, which yields a new axis called the first component. The second component is orthogonal to the first component and maximizes the variance of the projected data and so forth. The first and the second component span a plane which is called the most interesting plane, since it should provide the most information due to the maximized variance. This is not guaranteed, however. Also, a linear transformation cannot always produce a dimension-reduction: if for example, all points lie on a hypersphere - only nonlinear transformations can. Another point is that the PCA is extremely sensitive to outliers, mainly because distances between points are squared. Yehuda Koren and Liran Carmel [Koren2004] sighted these problems and suggested some enhancements. They introduced a weight for each distance and underweighted outliers, leading to a normalized PCA (see Figure 2.22).

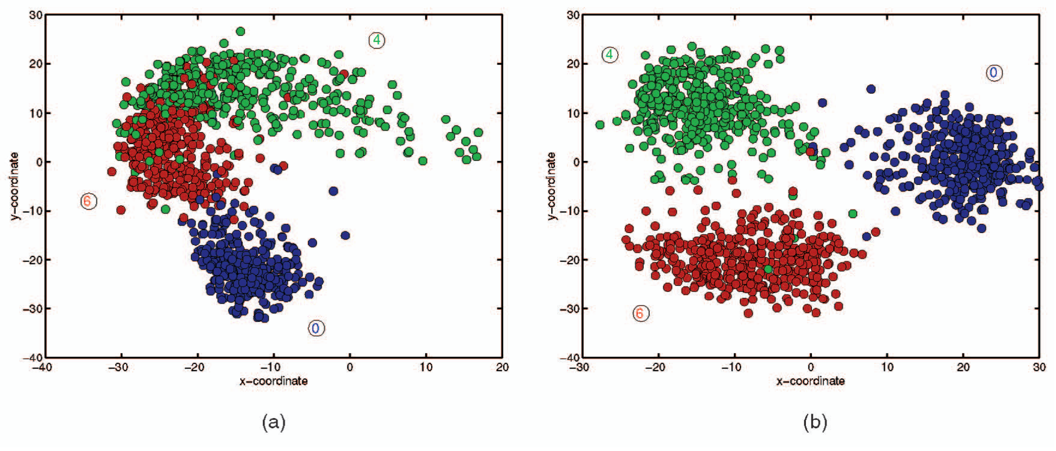


Figure 2.22: *Scatterplot using (a) PCA and (b) Normalized PCA of a 64-dimensional dataset[Koren2004]*

Projection Pursuit

The Projection Pursuit was developed by John W. Tukey and J. H. Friedman [Friedman1974]. The idea behind is, that the hyper room constructed by the multivariate data is projected on a hyperplane, and doing this for each possible projection with a linear mapping algorithm. Each of the hyperplanes is associated with an index, which shows how ‘interesting’ the structures are (interesting projections are aberrations to normal distributions). These indices are used to perform a heuristic search to locate the most interesting projection. The component along that projection is then removed form the original data, and the whole process will be repeated until no interesting structures are found anymore [Friedman1974]. In practise projection pursuit methods are often used for finding non-overlapping clusters.

The Grand Tour

In the grand tour [Asimov1985] projects the multidimensional dataset orthogonal on 2D-planes (scatterplots). It can be seen as a generalisation of rotations in high-dimensional space. Asimov’s idea was that to fully understand the nature of a data, one has to see them from all possible angles [Wegman2002]. Since there are infinitely many possibilities, they key is here to visit only a dense subset, meaning making a ’tour’ through those projections. The tour is chosen based on criteria like uniformity, density, continuity and even some form of user control [Elmqvist2008]. To achieve these goals, many algorithms are proposed, e.g. in [Wegman2002] or [Asimov1994]. Overall, this is similar to the projection pursuit method, but only low dimensional projections are visited.

2.3.5 Scatterplot Matrix

When we take only 2 axis of the mulltivariate input dataset at a time and map them to each other, all combinations of doing this yields nxn scatterplots, building a matrix [Cleveland1988] (see Figure 2.23). These scatterplots are then rendered beneath each other as *small multipes*, which were discussed in 2.1.3. Since mapping dimension i on dimension j will produce the same diagram as mapping j onto i (only rotated), and mapping j=i will only generate points along the diagonal, the number of useful plots is n*(n-1)/2 for an n-dimensional dataset. Generalizing this idea to other visualisations, we talk about a *spreadsheet approach* [Chi1997]. Unlike a normal spreadsheet, a visualization spreadsheet cell can contain "..an entire complex data set, selection criteria, viewing specifications, and other information needed for a full-fledged information visualization." [Chi1997]

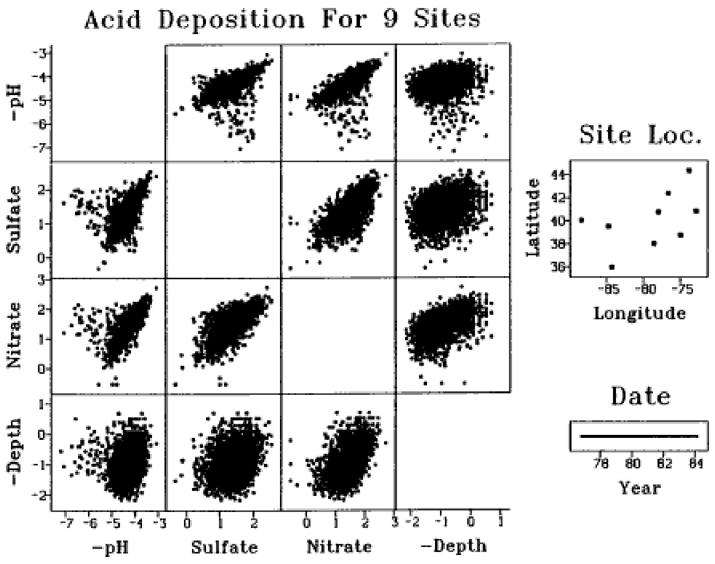


Figure 2.23: *A simple scatterplot matrix for 4-dimensional data. [Carr1987]*

While the saving of screenspace is obviously a big advantage of dimension reduction methods over the natural redundancy a scatter-matrix provides, one of the problems of dimension reductions methods are, that their solutions are often hard to understand. In his keynote *Visualizing Data for the Masses: Information Graphics at The New York Times* for the InfoVis 2007, Matt Ericson, Deputy Graphics Director at the New York Times said that they won't even use simple normal scatterplots in their paper, because their readers have issues understanding them - they expect time to be on the x-axis. Providing plots with linear-combinations of n dimensions mapped on 2 are far out of scope for the average audience [Elmqvist2008].

If the whole matrix is visualized, some problems arise with this approach. Because of the vast number of scatterplots that have to be drawn with higher order data-sets, issues like occlusion are emphasized here due to the small size (which just shrink to the size of thumbnails) each plot gets allocated [Voigt2002]. Using details on demand and focus+context techniques to overcome these issues are a central part of this thesis practical work. For example using the scatterplot matrix as an overview and requiring user action to retrieve further details is an often used approach. The detail view can then be a pop-up, showing a more sophisticated version of the chosen scatterplot. Or it can be embedded directly in the matrix, using distortion or using the space the natural redundancy opens.

The space a scatterplot-matrix provides in the diagonal (since no correlation-information could be retrieved when axis x = axis y) is commonly utilized by showing a histogram or the names for the corresponding row/column. Name-labels are also often shown directly beneath each row/column, as seen in Figure 2.23.

Dimensional Reordering

The user task while using a scatterplots-matrix is to find a projection with interesting patterns and structures. But because each dimension in the data-set can provide vastly different data, it may be quite difficult to see relations between projections. To improve this, we can just reorder the dimension in the matrix so that similar plots lie beneath each other, making comparisons much easier. Peng et al. therefore defined a clutter-measure [Peng2004] based on the cardinality each plot provides, which serves as a criteria for the dimensional reordering (see Figure 2.24).

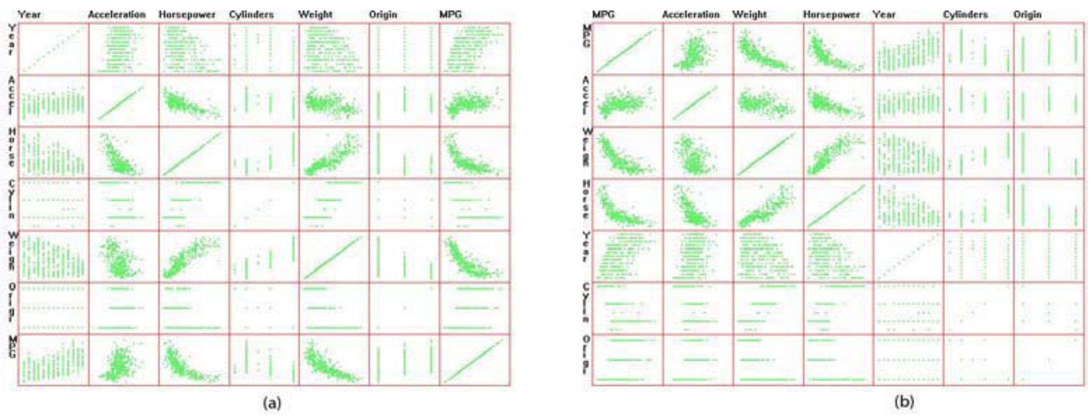


Figure 2.24: *Scatterplot matrix dimensional reordering. (a) shows the matrix with randomly distributed projections, dimensional reordering leads to (b) [Peng2004]*

Jigsaw map

Similar to dimension reordering, Yang et al. proposed a method, which is an approach especially suited for very high dimensional data [Yang2007]. They use a jigsaw map for producing the matrix grid, a technique for layout algorithms producing non-rectangular regions [Wattenberg2005], in which they embed in pre-calculated hierarchically ordered dimensions-projections of the matrix. The now visually grouped scatterplots helps when judging similarities between different plots (see Figure 2.25).

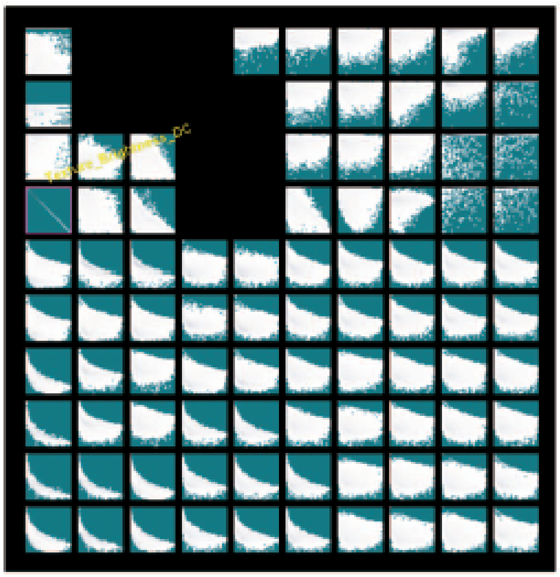


Figure 2.25: *Jigsaw map by [Yang2007]*

2.3.5.1 Navigation in the Scatterplot Matrix

As mentioned above, the scatterplot matrix is often just used for an overview in a focus+context system of the whole data set, letting the user navigate with the keyboard and/or mouse through it to select the desired axis for the main scatterplot-view. Without linking & brushing, just simple switching the main view lacks correlation and comparison between the different dimensions [Elmqvist2008]. Quad-vise plots are able to indicate correlations between 2 different scatterplots. Dimension-reordering techniques also help for comparison purposes on adjacent cells.

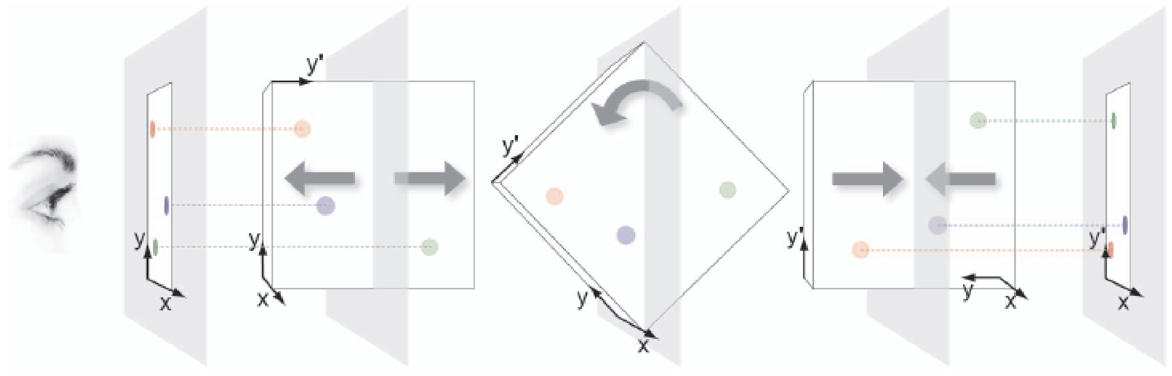


Figure 2.26: *Rolling the Dice:Scatterplot Matrix Navigation using the third dimension for transitions between different scatterplots [Elmqvist2008]*

Blending techniques or transition-animations of points while switching dimensions can also improve the perceived correlations between the dimensions. The grand tour can also be seen as *"invaluable tool for animating high-dimensional visualization"* [Wegman2002]. Elmqvist et al. proposed a new way to give the user more of a feeling of correlations while exploring the datatset with scatterpolts. They are using animated rotations and transitions in the 3D-Space between adjacent scatterplots, new introduced axis emerge from the third dimension. 'Hyperjumps' from one to another random plot yield to a chained animation series. This is illustrated in Figure 2.26.

2.3.6 Discussion

A scatterplot is a very commonly used visualisation tool, mostly due to its simplicity and its familiarity. Dealing with natural shortcomings like overplotting is the key for providing satisfying results with this visualisation technique. For displaying multivariate data, the decision must be made between dimension reduction methods or a spreadsheet approaches like a full-scale scatterplot matrix. While the former methods save screen space and some correlations in the data can be detected which may be missed otherwise, they are also limited by their predetermined chosen algorithm and jeopardized by wrong user decisions resulting of their higher complexity. For this reasons and the fact that scatterplots are not the only visualisation method in the Caleydo framework, we decided to go with the latter for the practical part of this thesis.

One will find only few statistic applications without a scatterplot feature, but also for the visualisation of gene-expression data, most frameworks like **GeneVAnD** [Hibbs2005], **Hierarchical Cluster Explorer** [Seo2002] or commercial tools like **Spotfire** support this type of visualisations.

Chapter 3

Concept

Caleydo, the visualisation framework on which this thesis practical work is built upon, already provides two common techniques for the visualisation of expression data, heatmaps and parallel coordinates (see Chapter 2.2).

We want to add another view based on (2-dimensional) scatterplots, using state-of-the-art methods and features. Also, a full integration of this view to allow a multiple-coordinated-views workflow and direct enhancements to the framework to make this happen are the main concepts of the practical part of this thesis.

Since the target-data consists of experiments witch each can contain thousand of gene expression data values, we gain a matrix of n*m (see Figure 3.1) data-values, that we want to visualize at once. Because a normal scatterplot allows us to only view the correlation between 2 experiments, we have to provide another method for that task: In our case, this is the scatterplot matrix (see Chapter3.2).

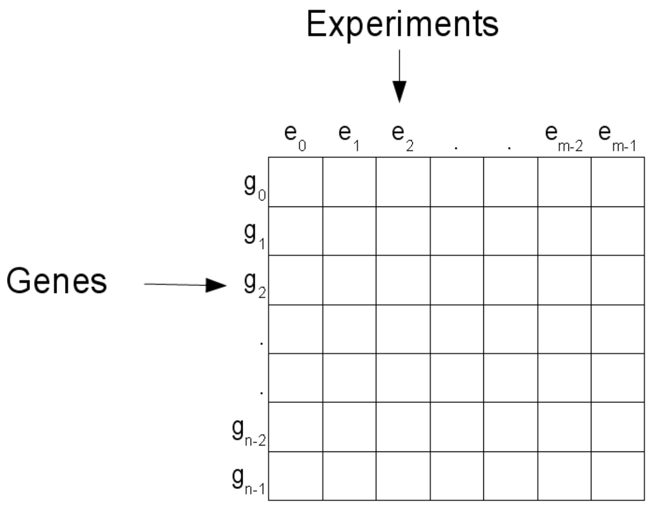


Figure 3.1: *Gene expression data in Caleydo: With n experiments and m genes we get a mxn matrix of data.[Schlegl2009]*

3.1 The main Scatterplot-view

Our scatterplot should provide at least the following visualisation features and requirement:

- **Performance :** On a standard system, each interactive operation should be able to be performed in realtime. This should be accomplished by using layered rendering and (if necessary) by random sampling (see chapter 2.1.5.
- **Different Representation of Points :** Scatterpoints visualisation should be able to customize, the user should be allowed to chose from different point-primitives. Pointsize should also be scalable. Transparency should be supported as well.
- **Scalable Coordinate System :** A two axis coordinate system should be drawn depending on the logarithmic or linear scale, provided from the chosen input data-scale of the Caleydo framework. Axis (experiments) should be labeled.
- **Mouse Over Detail on Demand :** Moving a mouse over a scatter point should reveal additional detail information (value/name/status..).
- **Support of quad-vise plots :** Two experiments should be selectable for one axis, using different color-coding. They also could be connected with lines, to easily see their correlation (see chapter 2.3.2).
- **Zoom Feature :** Because there may be a lot of points (currently over 20k) in one scatterplot which also may be heavy clustered, a zoom-feature for the main view could be helpful.

3.2 The Scatterplot Matrix

Beside dimension reduction methods, the other method for showing multivariate data with scatterplots is the spreadsheet approach called scatterplot-matrix (see Chapter 2.3.5). We decided to go with this direction instead of the former for the following reasons: First, dimension reduction methods are far more complicated to understand for the average user [Elmqvist2008]. Second, with methods Caleydo already provides, as well as with the help of new features like quad-vice plots or the new support of multiple brushes, there are tools available for displaying correlation between more then two experiments.

Since with this approach the number of visualized scatterplots rises exponentially, we need to be very careful on how we deal with this now huge number of data-values. A rather simplified visualisation of the scattermatrix cells are here to be expected, which may not even have a negative visual impact due to the size of the matrix cells.

3.3 Focus plus Context

As we want to view all experiments at once without losing the ability on concentrating on specific genes, we need powerful focus plus context techniques to accomplish this. We discussed this already in the related work section under 2.3.5. Since the scatterplot matrix already provides us with an overview and the main scatterplot with the focus-part, we only need to merge them together into one view. One of the possibilities we have here is to develop both views separately, and let the user arrange his focus plus context system directly via the framework.

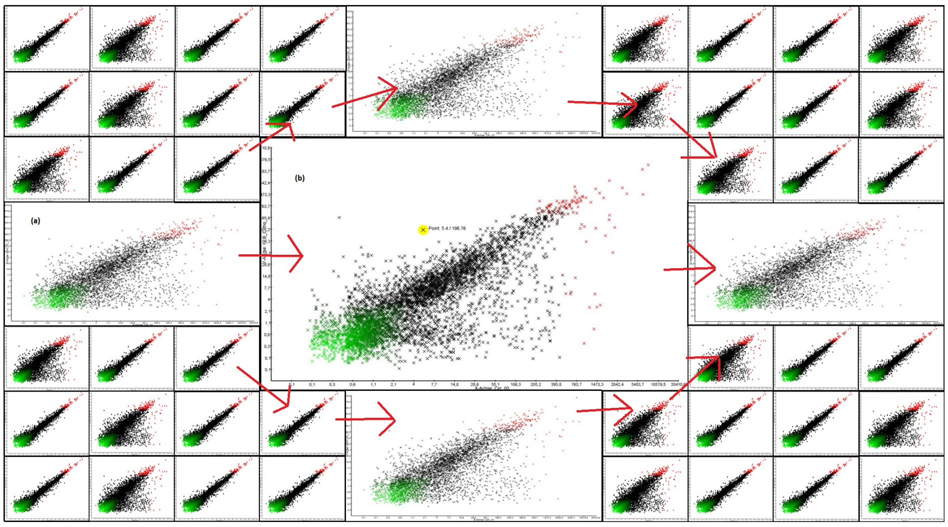


Figure 3.2: *Concept of the embedded main-view in the scatterplot matrix. While the more detailed, bigger versions of the scatterplot cells follow the orthogonal order of the matrix, the smaller plots do not. The arrows show where cells will move when cell (a) replaces cell (b) as main focus.*

The other possibility is to embed the matrix directly into the main view. Early design sketches of this idea are shown in Figure 3.2 and Figure 3.3. While using a distortion based fisheye-method as shown in Figure 3.3 has the disadvantage of distortion even in the main view, Figure 3.2 loses the orthogonal order of the rows and columns in the matrix. Also, if we want to visualize the whole matrix, we get weird empty spaces on the edges of the matrix.

For this reasons, we decided to follow another, two-way approach: We embed the main scatterplot view in the space the matrix' natural redundancy opens in the right upper part (we render only half the matrix). And second, we allow an additional, on demand zoom directly in the matrix to provide an intermediate detail level between a matrix cell and a main view. This allows the user to easily search for an interesting scatterplot in the data while still maintaining full detail on the main view without wasting the valuable screenspace we have at our disposal.

3.4 Linking and Brushing

When explaining linking and brushing to someone, this is very often done by showing a picture of a scatterplot matrix doing that. (see Section , Figure 2.8). So naturally we provide this technique here, too. Of course, brushed data should be propagated not only to the matrix cells, but to the other available views as well (and vice-versa).

Clicking points for selecting and deselecting specific points, mouse-over highlighting and brushing multiple points in the scatterplot with a rectangular brushing tool are needed. Multiple brushes at once have to be supported.

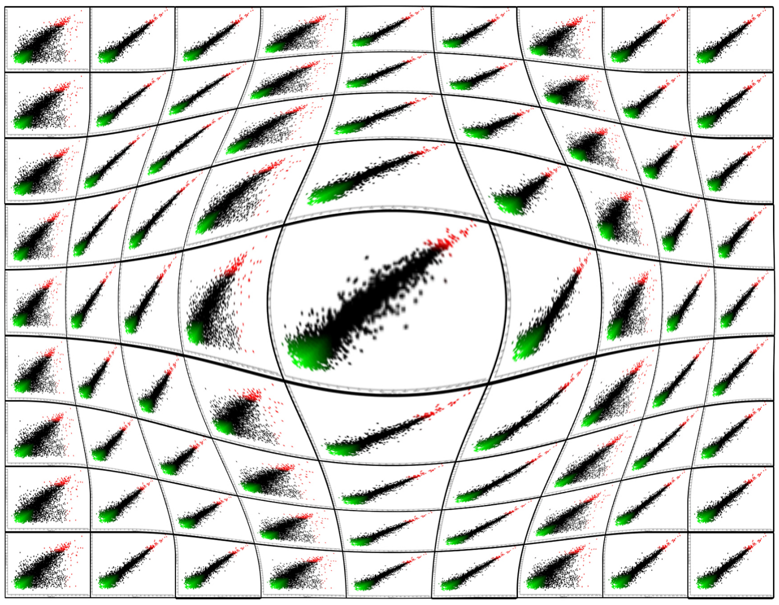


Figure 3.3: *Concept sketch of scatterplot matrix using fish-eye distortion*

3.5 Zooming in the Scatterplot

While the focus view can provide us with a good view on the details, points in there can still be cluttered. This actually does not solely depend on the amount of points we have to visualize, but rather their distribution. Especially for picking purposes, even a very low amount of points can make selecting a single gene an impossible task if their coordinates are very similar. Therefore, we need to provide a zooming technique in the main view as well. For this, we follow an orthogonal stretching approach, where we let the user drag the bar handles in realtime.

3.6 Integration into the framework

The scatterplot view should be an integrated part of future Calyedo builds. Therefore it should inherit its common concept, technically and visually. For example, scatterpoints use the (optional) same color-coded gene-expressions like in the heatmap, the common toolbar, the random sampling methods; custom brushes should not be bounded to a single view. This also means, that new methods developed especially for the scatterplot view like multiple brushes have to be made available for the other views as well. For this reason, we provide another view for managing selected data, which allow an overview and some information of the selections and operations like merging or deleting.

Chapter 4

Design and Implementation

In this chapter we focus on the implementation details and take a look at the design challenges of this thesis' practical part. In Section 4.1 we take a look at the available technologies used for our work, Section 4.2 gives a brief overview of the architecture on which the Caleydo framework is built upon. Finally, we discuss the aspects directly related to the scatterplot implementation in Section 4.3.

4.1 Used Technologies

Since this work runs as integrated view in the Caleydo[1] framework, we have to deal with the technological environment the framework is based on. Non-technical aspects of the framework are discussed in chapter 2.2.

Java

Caleydo is written completely in Java[2], a general purpose, cross-platform object-oriented programming language. Java is an interpreter language and runs in a runtime environment called *Java Virtual Machine (JVM)*. Caleydo runs under the current version 1.6.

Eclipse and the RCP

Eclipse [3] is the open-source *IDE (integrated development environment)* on which the development process of Caleydo happens. It was written in Java and was originally meant to be a *Software Development Kit (SDK)* for Java only, but in the meantime additional languages like C/C++ or COBOL are supported via the plug-in architecture. In Eclipse, most features are provided via plug-ins, making it a very customizable environment. The version used for this work takes the number 3.5 (Galileo).

Derived from Eclipse, the *Rich Client Platform (RCP)[4]* provides the user with tools to built a rich client application. In contrast to fat clients, rich clients can enhance the functionality with plug-ins to solve even foreign problems.

[1]http://www.caleydo.org/
[2]http://java.sun.com/
[3]http://www.eclipse.org/
[4]http://wiki.eclipse.org/index.php/Rich Client Platform

SWT

One part of the RCP is the *Standard Widget Toolkit (SWT)*[5]. It provides access to GUI elements like sliders, buttons, toolbars, menus and so on. The SWT is available for a lot of platforms, but since the toolkit uses the native GUI-APIs each operating system provides, it has to be implemented for each platform separately, which results in divergent performances, and on some platforms it is not available at all.

Jogl

In Caleydo, the SWT is mostly used for the menus, toolbars and small views like the selection browser. For the main visualisation part, it uses OpenGL[6] via Java Bindings for OpenGL (JOGL)[7]. JOGL (Caleydo uses the 2.0 Version at this time) is a wrapper for using native OpenGL commands directly in Java.

4.2 Framework

Caleydo's purpose is not only the allocation of several InfoVis methods and techniques, but also for rapid and easy development of new ideas in this area [Lex2008]. Therefore, Caleydo provides us with a plethora of classes to let the developer rather focus on his ideas then on tiresome technical details. The architecture of Caleydo is already thoroughly discussed in [Lex2010]. To avoid redundancy, we describe here only briefly the concepts used especially for this work and new enhancements like the plug-in architecture.

4.2.1 View Management

Caleydo supports two types of views: SWT views and OpenGL views.

SWT Views

SWT views are used for simple visualisations, using mostly SWT-functionality. The newly added `SelectionBrowser` is such a view, it contains SWT buttons and a tree-widgets for the management of selections (see section 4.3.4).

OpenGL Views

OpenGL views are used for more complex visualisations. OpenGL provides much more flexibility and the ability to use the 3rd dimension. All OpenGL views extend the abstract `AGLView` class, which provides basic functionality like picking and basic initialisation. The in this work implemented `GLScatterPlot` class is a OpenGL view, an extension of `AStorageBasedView` (which itself extends `AGLView`), which provided additional tools for views heavily based on table-like data (heatmaps, parallel coordinates). An additional advantage of Caleydos OpenGL views are that the support remote rendering, which means that on

[5]http://www.eclipse.org/swt/
[6]http://www.opengl.org/
[7]https://jogl.dev.java.net/

view can be rendered inside another, which is demonstrated in the **bucket** [Streit2007] or the **hierarchical heat map** [Schlegl2009].

4.2.2 Plug-In Architecture

During the development of this thesis, a paradigm shift happened regarding the availability of views in this work. Each main view is loaded via plug-in now, so the desired complexity of Caleydo may now vary depending on its configuration. Each loaded view can provide its own toolbar containing SWT widgets, renderstyles, special event-listeners and so on. The added flexibility with this approach makes the development of new, experimental methods a very easy task without interfering too much with the code in the main framework.

4.2.3 Storage Concept

The Caleydo framework deals with the data using a storage concept introduced by Michael Kalkusch [Kalkusch2006] and further developed by Alexander Lex [**?**].

The top layer in the storage concept are the **UseCases**, which holds all information of the specific data and are used as interface between the views and the data [Schlegl2009]. UseCases can hold **Sets**, which are the containers of the data itself, containing multiple **Storages**. Storages are simple arrays that hold the raw data. The array type can hold various types of data like strings or floats and even objects. The gained performance advantages of arrays are paid with their inflexibility. For this reason they are accessed only via **Virtual Arrays**, which consist solely of indices to the data in the storage-arrays. Operations like reordering of the data (which is needed in clustering algorithms) therefore happen directly in the virtual arrays, letting the raw data in the storage-arrays completely untouched. The `Set` respectively `Storage` class contain a lot more methods then the access to the data, like normalisations or conversions to a logarithmic scale.

4.2.4 Event System

Communication between views in Caleydo is event driven. The event system is designed following the Gang of Four [Gamma1995] *Observer* and *Mediator* Design Patterns [Lex2008]. The central class is the `EventPublisher` which handles the events. A class can now trigger an event with the `EventPublisher` class, which propagates the event to all receiving classes. To receive an event, a class has to subscribe an instance of the listener interface **AEventListener** to the `EventPublisher`. The `handleEvent()` method of the listener class executes the event then. This should be only done in the same thread, otherwise there exist the `queueEvent()` method which guarantees thread safe execution.

4.2.5 Selection Management

The selection mangers are based on the `VABasedSelectionManager` respectively `SelectionManager` which store sub-selections of data-items (not directly, their indices via Virtual Arrays) in hash maps. Since selection play a big role in this work, they are discussed in more detail in Section 4.3.4 .

4.3 Software Design

The main class this implementation relies upon is the GLScatterPlot View derived from `AStorageBasedView`. The class is supported by the static class ScatterPlotRenderStyle, which is responsible for graphical configuration like z-values for the layers, spacing, initial values and so on.

Figure 4.1 shows the UML Diagram of the scatterplot-view class.

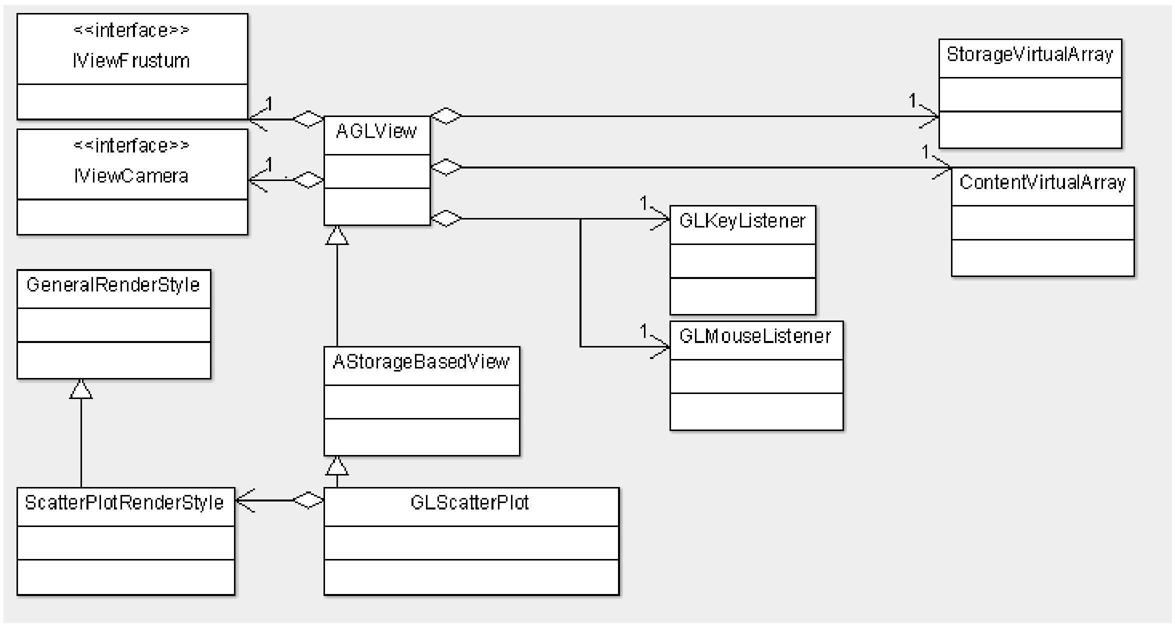


Figure 4.1: *UML Diagram of the Scatterplot Class*

4.3.1 Display Lists

For optimizing performance we use the layered rendering approach we discussed in Chapter 2.1.5. For the rendering-part we use the OpenGL *Display Lists*. Display Lists group up OpenGl commands and store them in the graphic card memory, which, when invoked, can be executed at once in the order they were listed. So for static data, they are a huge performance gain, since the objects created by the list need to be computed only once. Their disadvantage come with the fact that they cannot be changed once built without completely recomputing them, and not all OpenGL commands are allowed. They also may raise compatibility issues with some calls, for example the `TextRenderer` class in JOGL has issues with multiple display lists.

Layer	Init View	Resize Window	Scale Glyphs	Update Colour Coding	Update Selections (Mouse Over)	Update Selections (Brushes)	Toggle Zoom	Toggle Normal<->Quad-vise	Use Matrix Zoom	Change Experiments	Toggle Matrix<->Main
Mouse Over	yes	yes	no	no	yes	no	yes	yes	yes	yes	yes
Coordinate System	yes	yes	no	no	no	no	yes	no	no	no	yes
Render Matrix Full	yes*	yes*	no	yes*	no	no	no	no	yes*	no	yes*
Render Matrix Sel.	yes*	yes*	no	no	no	yes*	no	no	no	no	yes*
Scatter Points	yes	yes	yes	yes	no	no	yes	yes	yes	yes	yes
Render Selections	yes	yes	yes	no	no	yes	yes	yes	yes	yes	yes
Calculate Matrix Sel.	yes	no	no	no	no	yes	no	no	no	no	no
Calculate Matrix Full	yes	no	no	yes	no	no	no	no	no	no	no

Table 4.1: *Scatterplot Layers. (*): those operations only occur while in Matrix View Mode.*

For this reasons, we try to set up logical layers that need to be updated as rarely as possible. Table 4.1 shows the display lists and needed calculations in the rows and the events or actions that cause rendering updates as columns. The different layers vary vastly in their performance costs, and by design we tried to accomplish a reciprocal correlation between update frequency and execution time. In Table 4.1 we see that the most expensive operation (calculate the entire scatterplot matrix) needs to be only performed in two occasions: on startup and when we change the color-coding of the data items.

4.3.2 Sampling and Textures

The obvious approach for rendering the scatterplot matrix is to make use of Caleydo's ability to draw nested views by using the remote rendering feature (see Section 4.1). This is not feasible for two reasons. First, even when we allow the scatterplot to be rendered in an extremely low detail mode, the space-allocation between the main view and a matrix cell differ extremely. For a typical screen resolution of 1 million pixels for the main scatterplot view (substracted are toolbars, menus and borders), with 50 experiments we have only 400 pixel per matrix cell. This is not only a waste of rendering time for no visual gain, without sophisticated pixel-allocation in the cell we would get very bad results anyway (a 30.000 points scatterplot rendered with 400 pixel would result in a solid square without any means of density, even if the plot shows strong correlations).

Second, like discussed in the above section, we want to minimize the actual re-rendering. With a simple usage of nested views, a view resizing would cause a full recalculation of the whole matrix. An intermediate stage would be necessary, for example the nested views are rendered in a back-buffer and then copied into textures for their visualisation. Since the rendering of one texture is trivial instead of a whole view in regards to performance, the texture could be used as substitute of the whole view and only needs to be re-rendered on special occasions, as shown in Table 4.1.

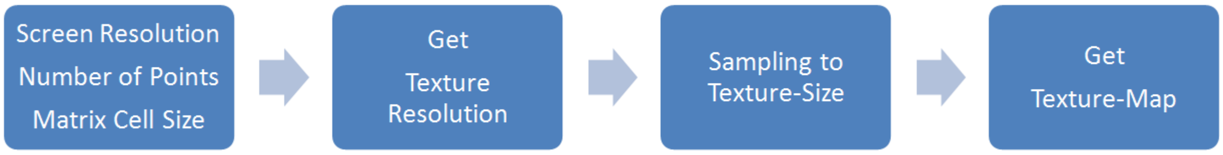


Figure 4.2: *Diagram shows the process for getting the texture of a matrix cell.*

For this reasons we implemented the matrix visualisation by rendering each cell directly into texture. This lead us, beside the performance gain, to an additional advantage: The down-sampling to the texture resolution is actually the same process as the binning-technique to solve overplotting issues (see Section 2.3.1). Figure 4.2 shows the process in detail: First, with the information gained from the available screen resolution and the actual matrix cell size we get form the number of experiments, we determine the actual texture resolution. Then we scale the input data to the texture size. We then write the data directly into the array the texture-map spans, taking Caleydo's predefined random sampling rate into account. Since for some texels there are naturally more corresponding data-points, we use an alpha value to indicate the density in this region. The textures for brushed data in the matrix are determined in the exact same way, only with a slightly lower resolution for a better recognition of small selections. When visualised, both textured are rendered above each other using transparency.

4.3.3 Zooming

For the zooming in the matrix, we just stretch the texture under the mouse-cursor by a certain factor to gain the desired results. Since we don't allow any distortion, the position of all the textured surrounded by the highlighted cell have to be adjusted as well. This results in a variance of space for the embedded view which tries to utilize as much space as possible. So when the mouse is hovered across the matrix, this forces often a recalculating of the three depended display lists (see Listening 4.1).

Listing 4.1: Transfer function for orthogonal stretching

```
private float transformOnXZoom(float x) {
                if (!bMainViewZoom)
                        return x;

                if (x < fTransformOldMinX) {
                        float factor = fTransformOldMinX / fTransformNewMinX;
                        return x / factor;
                }

                if (x > fTransformOldMaxX) {

                        float factor = (1 - fTransformOldMaxX) / (1 - fTransformNewMaxX);
                        return fTransformNewMaxX + (x - fTransformOldMaxX) / factor;
                }

                float factor = (fTransformNewMaxX - fTransformNewMinX)
                                / (fTransformOldMaxX - fTransformOldMinX);
                return (fTransformNewMinX) + (x - fTransformOldMinX) * factor;
        }
```

For the zooming in the main view based on orthogonal stretching, we use a transfer function for the calculation of each xy-value. Each graphic rendered, regardless if they are the main scatterpoints, the brushed data or the coordinate system, has to be adjusted by its position relative to the zooming-window. The code-snippets shown in Listening 4.1 and 4.2 show the transfer function for the x coordinate. The `fTransformOldMinX` and `fTransformNewMinX` respectively `fTransformOldMaxX` and `fTransformNewMaxX` indicate the new borders of the zoom window and their initial positions.

Listing 4.2: Overloaded transfer function with added scaling and position adjustments

```
private float transformOnXZoom(float x, float fSize, float fOffset) {
                float tmp = (x - fOffset) / fSize;
                return transformOnXZoom(tmp) * fSize + fOffset;
        }
```

The method in 4.2 is used when the position and scale of the main view does not match with the frustrum of the whole view (like in the embedded view mode).

4.3.4 Selection Management and Brushing

For brushing across multiple views, classes that store and manage those data are a necessity. Like already indicated in Section 4.2, the Caleydo framework accomplishes this by using the `SelectionManager` class. Figure 4.3 shows the UML diagram of the class and the related ones.

For storage-based views, two `SelectionManagers` are common: The `contentSelection-Manager` are used to handle the genes while the `storageSelectionManager` is responsible for the experiments.

Selections can be of four distinct default types, defined by the `SelectionType` class: NORMAL, MOUSEOVER, SELECTION and DESELECTED. The `SelectionType` defines the name, color, priority (used mostly to determine consistent z-values when selections are rendered above each other) and other graphical and functional properties. NORMAL types define here the base type, where all items are initial stored. Another key feature is the `SelectionDelta` which is used to propagate changes in the selections to other views. It is reset every time it's getter is called.

The `SelectionManager` class provides many methods for moving the selected elements in and out of the storage or between different types. In older Calyedo versions, the selection

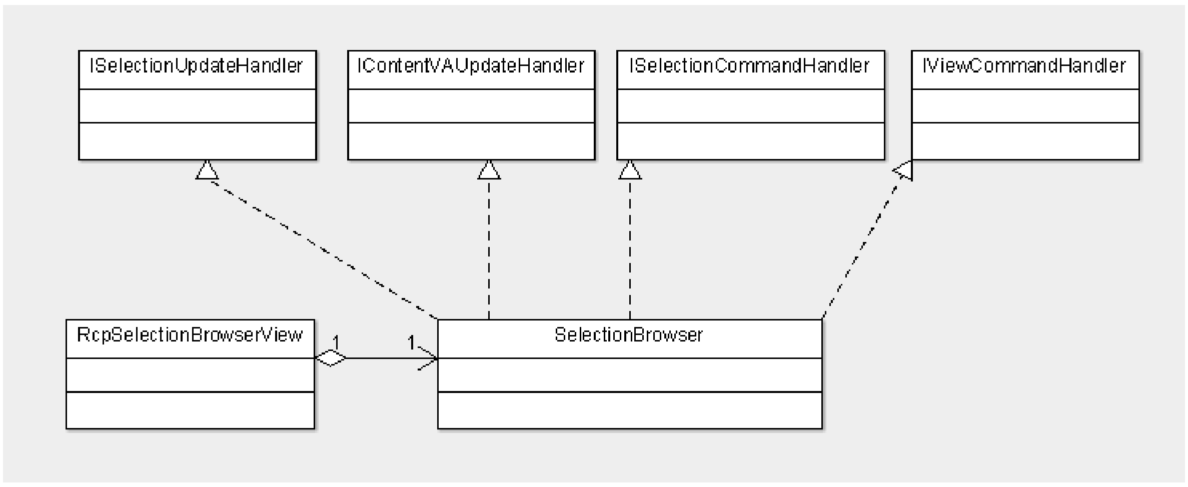


Figure 4.3: *UML Diagram of the SelectionManger class*

types where mutually exclusive and unique. The design of the scatterplot view demanded a removal of these restrictions though. For example, there was no possibility to have the same elements as NORMAL and MOUSEOVER selections in the same `SelectionManager`. Also, the ability to add an arbitrary number user-defined selection types has been added, since we wanted to support multiple brushes.

Brushing in the Scatterplot View

For the default selection-types, the `PickingManager` handles these elements. Since the technique is very similar how it is implemented with heatmaps and parallel coordinates, we will not go into more detail here, [Lex2008] already covers this topic.

To add a new custom brush, the user has to simple draw the brushing primitive over the visualized data. Currently, only the rectangular brush is supported, but it would be easy to add other types of brushes. So in this case, the rectangle is drawn by dragging the mouse over the desired data-points. Once the mouse-button is released, a new `SelectionType` is created with a predetermined color and a name consisting of an abbreviation of the view and the number of the current brush. The priority is set slightly higher than the other brushes which may already exist, since we do not want the newest brush to be covered by older ones. After that, an in-out evaluator method is executed which determines if points are inside or outside the brush-borders. Elements inside are added with the `contentSelectionManager` to the new custom selection. Then a event is created with the new obtained `SelectionDelta`. This `SelectionUpdateEvent` is then triggered with the help of the `eventPublisher`. So every view who listens to this event can then update its visualisation accordingly. The scatterplot view itself listens to this event as well and sets two flags that, on the next rendering cycle, cause the display-lists responsible for the selections to be recalculated.

The Selection Browser

Since we implemented the support of multiple brushes, we want to allow the user itself to have some more control over the different selections. Therefore we added a new SWT-view for managing them. The view uses the **Tree** widget for visualisation of the custom brushes and the **button** widgets to activate the actions like merging and deleting selections. The `SelectionBrowser` represent the GUI which invokes the appropriate methods of the `contentSelectionManager` class. It is also responsible that the `SelectionUpdateEvent` is triggered with the corresponding `SelectionDelta`, which leads to the results described above.

Chapter 5

Results

This chapter discusses the outcomes obtained out of the concepts and design-decisions described in Chapters 3 and 4. We first show the implemented features from the main-view in Section 5.1, discuss the scatterplot matrix execution in Section 5.2 and then take a look on the brushing and selection enhancements in the framework (Section 5.3).

Figure 5.1 shows an overview describing most features this view provides. We will discuss each feature more thoroughly below. The scatterplot view is available through the *Views* menu if the current configuration has loaded the scatterplot view.

Since we only focus on the visualisation of gene-expression data, we talk about experiments for the dimensions and genes for the actual data-values. Unless said otherwise, the screenshots in this chapter show human genome test-data with 12344 genes and 14 experiments.

Navigation happens via mouse or keyboard. Every feature is accessible via toolbars, direct mouse-interaction or keyboard shortcuts (see Appendix A). General features like resetting selections are available via the general toolbar, the scatterplot-specific features can be accessed via the scatterplot toolbar.

5.1 Scattterplot Main View

Regardless if embedded in the matrix or not, in both views everything works the same with one major difference: In the embedded view, changing the experiments with the cursor requires a confirming ENTER, in the mainview-only mode changing experiments happens with CURSOR-keys only.

Basic Visualisation

The scatterplot main view provides a coordinate system which can be switched off in case it occludes the direct view on the data. The coordinate systems' scale is chosen by the value-range the input data provides, so that the whole base data fits in the plot and uses as much space as possible.

The data point glyphs are color coded using the Caleydo framework color-coding, which can be changed on the fly with Caleydo's histogram view. Since we have 2 dimensions instead of the one that the histogram delivers, we take the maximum value of each axis to determine the color. The color-coding of the glyphs can be turned off (rendering them black only) as well, which is useful for easily distinguishing brushes from the rest of the data.

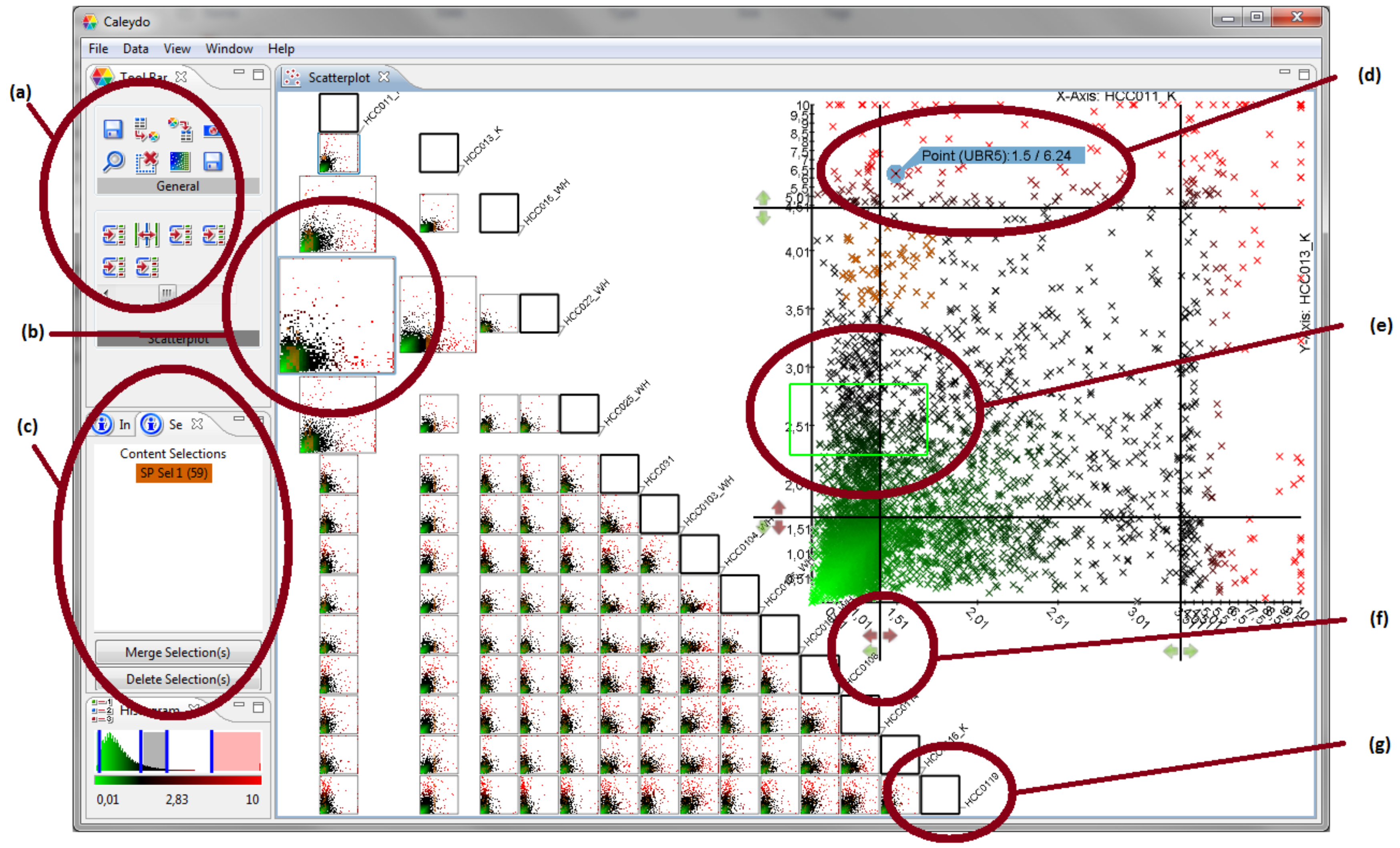


Figure 5.1: *An overview with most features enabled in the scatterplot implementation: (a) toolbar, (b) matrix-zoom, (c) selection browser, (d) mouser-over selection, (e) rectangular brush, (f) mainview-zoom sliders, (g) experiment labels*

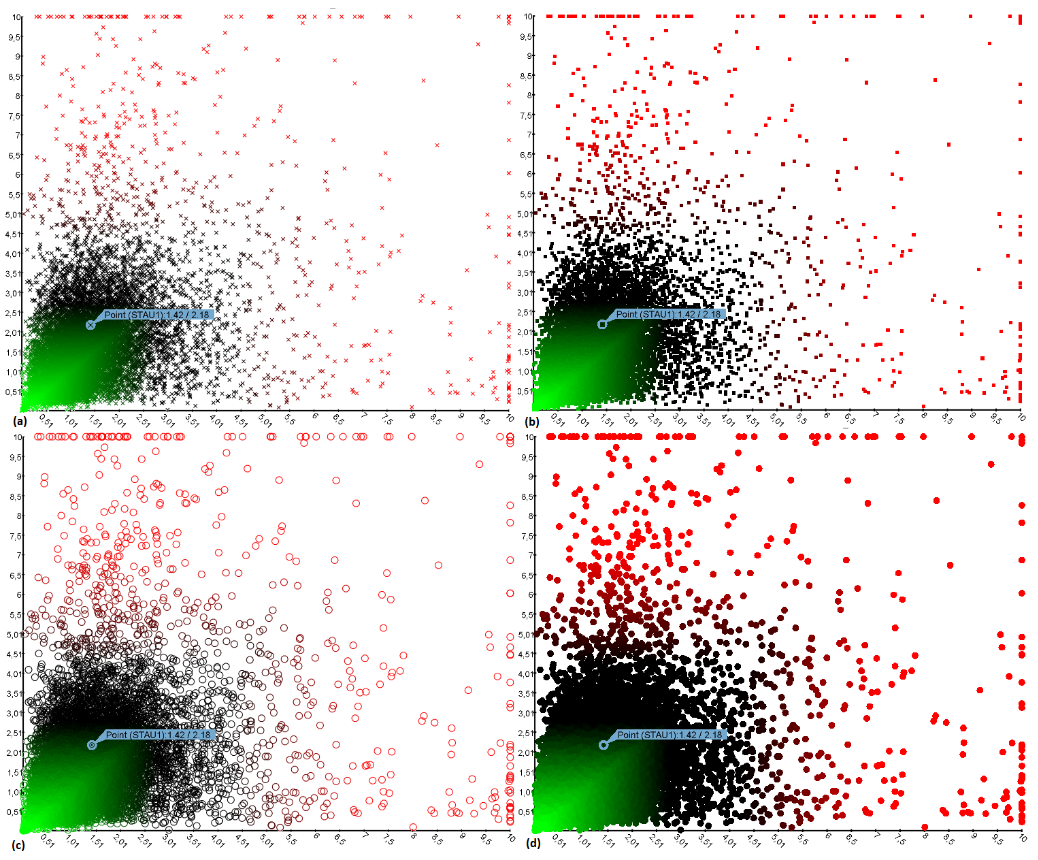


Figure 5.2: *The Caleydo scatterplot supports different point-primitives: (a) Cross - (b) Box - (c) Circle - (d) Disk*

The plot supports many different glyphs, some are shown in Figure 5.2. The glyphs are also interactive scalable with a slider-widget, as shown in Figure 5.3. Since the selected pointsize will mostly depend on the size of the input data, the general rule also is that bigger points provide an easier picking for selections like mouse-over details for particular genes, lower scaled points are less affected from overplotting issues but suffer from the fact that outliers are hardly visible (see Section 2.3.1).

The Zoom Feature

To further unburden the overplotting issues and to help with easier picking, the scatterplot main view provides a zooming-feature (see Figure 5.4) based on orthogonal stretching. Four bar handles are provided for each axis. Picture (a) shows the undistorted source image, only two bar handles are visible per axis. The red area between the bar handles shows the part which will be stretched. Clicking on the red arrow-glyphs let us change this source-area. Clicking on the green glyphs will allow us now to stretch the glyph at will (indicated by the green area), as seen in picture (b) and (c). If the picture is already stretched in any way, the base (red) area cannot be changed any more unless the stretching is reversed. This is to not confuse the untrained user. Picture (d) finally shows the interaction with quadvise-plots and the zoom feature (here only a selection of the data containing only points inside the stretched area are shown).

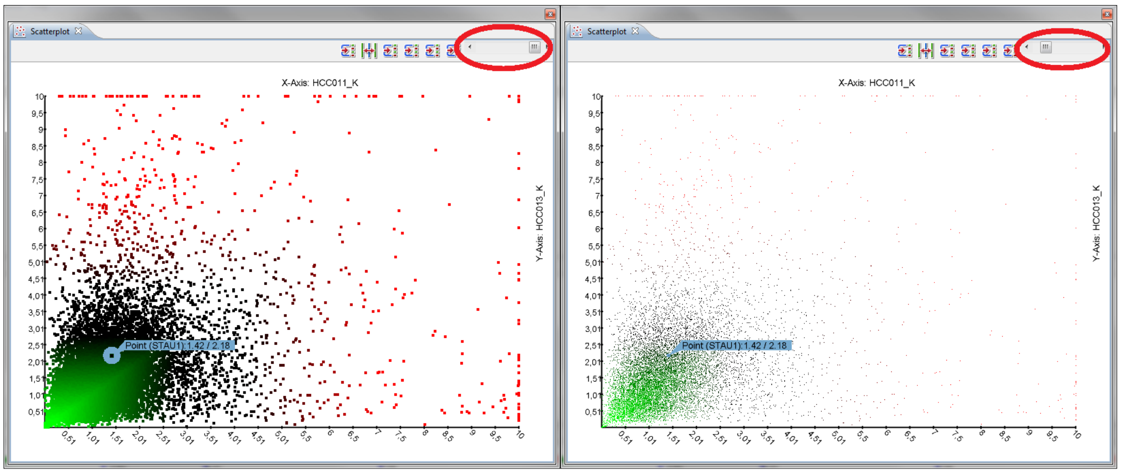


Figure 5.3: *The scatterplot point sizes are adjustable (see slider in red circle). While small pointsizes like in the left image give a better feeling for density distribution, outliers are really hard to see.*

We believe that this distortion based zoom feature is best suited for the tasks it needs to accomplish: Easier picking in heavy cluttered areas and revealing trends in small local areas. This technique has advantages over fisheye-zooms and simple pane and zooms techniques (See Chapter 2.1.2.1): No distortion in the focus area and no occlusion of any data-points. The additional visual hint accomplished with the red-tinted area to indicate the source-area also greatly helps with orientation and enriches the usability.

Performance and Random Sampling

Despite the fact that the performance for the test data seems sufficient even for a low-end development systems like the one used for this thesis (Intel Pentium 4 3.2GHz, 2 GB RAM, NVIDIA GeForce 7800 GS 256 MB,Windows 7), the operations for updating the whole scatterplot matrix still stresses the system just from pure numbers (See Section 4.3). Table 5.1 shows the delay for this operation with different number of genes. Since the runtime values are not exactly reproducible, we took 3 experiments:

#genes	Test 1	Test 2	Test 3
514	375 [ms]	343 [ms]	390 [ms]
6172	4109 [ms]	4785 [ms]	4609 [ms]
12344	12656 [ms]	12641 [ms]	12625 [ms]

Table 5.1: *The time it takes for re-rendering the whole scatterplot matrix is linear depended on the number of genes.*

Therefore, the scatterplot view fully supports the Caleydo random sampling feature. Picture 5.5 shows the three versions of the same data with different sample-rates. While picture (a) shows the entire data, in picture (b) the number of genes is cut in half (around 6000 genes). In picture (c) the number of points are greatly reduces to around 500. While (b) does not suffer much form the downsampling, picture (c) is not able any more to show

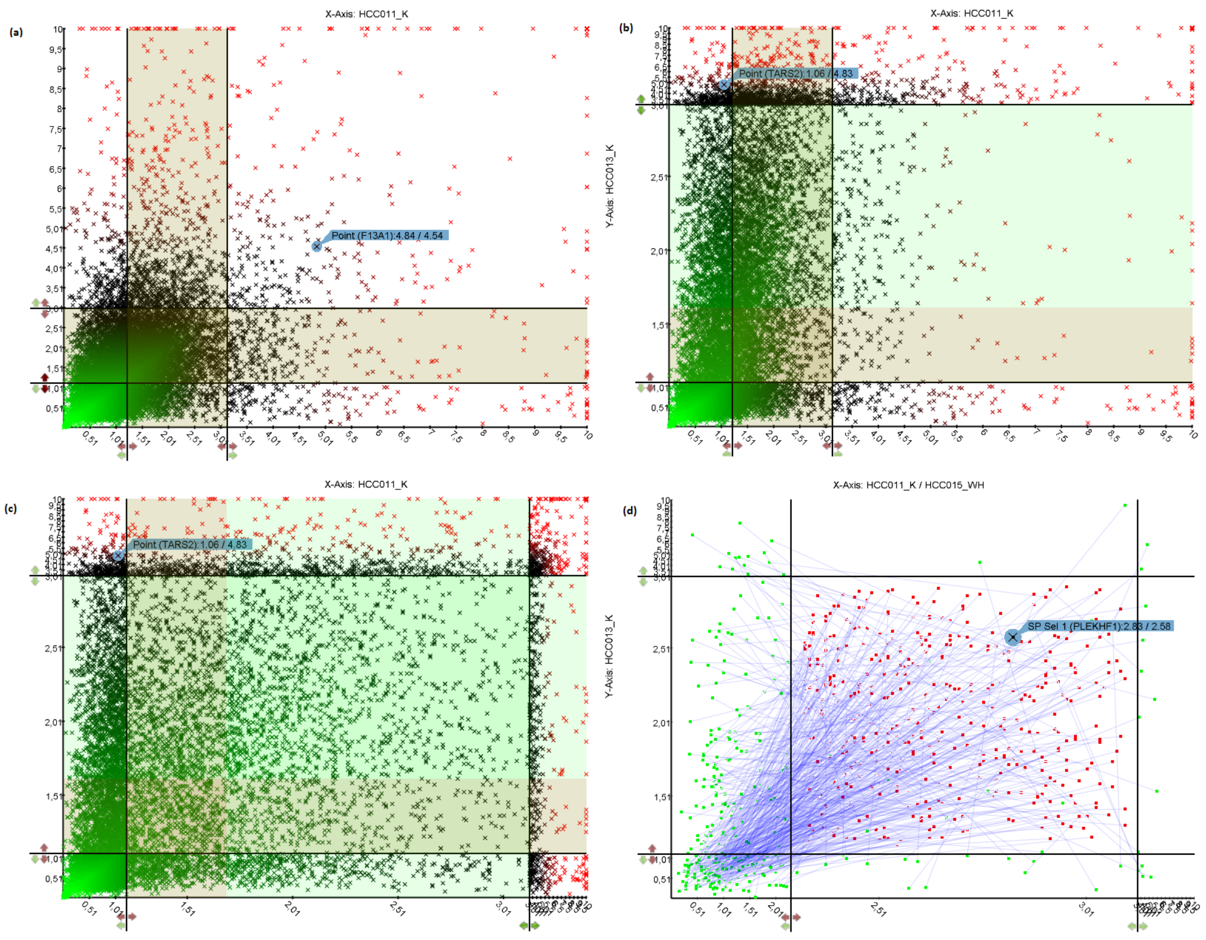
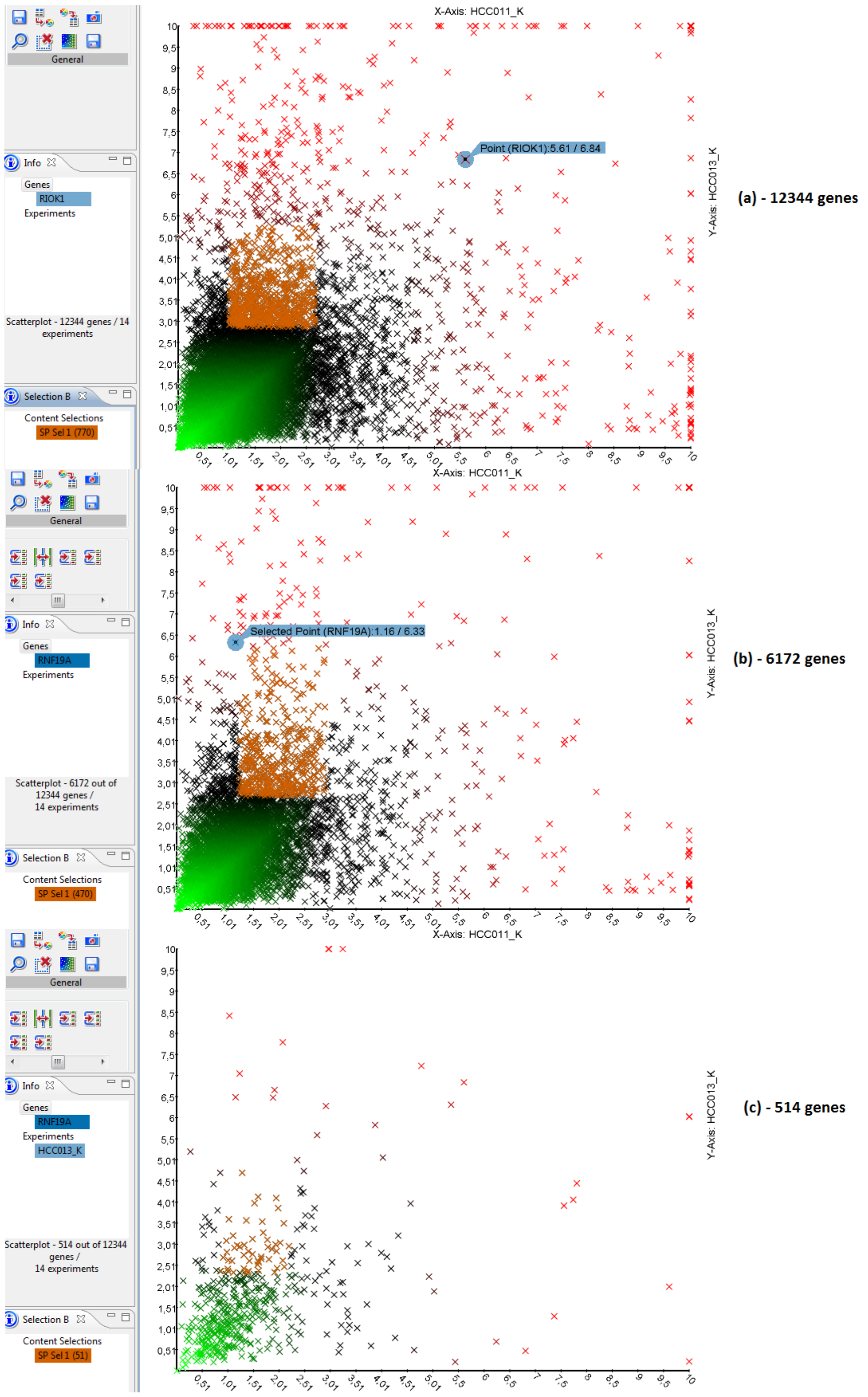


Figure 5.4: *Zooming using orthogonal stretching: (a) the source with 4 bar handles, (b) stretched in the y-direction (indicated by the green area) (c) stretched in x- and y direction. Picture (d) shows the interaction with quadvise-plots, showing a selection equivalent to the source area.*

the density distribution in the scatterplot. It still can provide an overview of general trends in the data, though.

Since performance is linear depended on the number of genes to be visualized, the required sampling rate for acceptable response time should be trivial to set.



General
Info
Genes
RIOK1
Experiments
Scatterplot - 12344 genes / 14 experiments
Selection B
Content Selections
SP Sel 1 (770)
X-Axis: HCC011_K
Y-Axis: HCC013_K
Point (RIOK1):5.61 / 6.84
(a) - 12344 genes
RNF19A
Scatterplot - 6172 out of 12344 genes / 14 experiments
SP Sel 1 (470)
Selected Point (RNF19A):1.16 / 6.33
(b) - 6172 genes
HCC013_K
Scatterplot - 514 out of 12344 genes / 14 experiments
SP Sel 1 (51)
(c) - 514 genes

Figure 5.5: *Scatterplot Random Sampling. While there is not much difference between picture (a) and (b), the greatly reduced number of samples in (c) loses most outliers, but still can indicate general trends in data distribution.*

Quadvise Plots

As discussed in Section 2.3.2 quad-vise plots show the correlations between two pairs of variables. Picture 5.6 shows two versions of it. (a) shows the correlation between 2 pairs of experiments, while (b) shows the correlation between a pair of experiments and an additional experiment, resulting in lines parallel to the x-axis. Lines are drawn with transparency for better results with a high amount of visualized genes. Since for example with our test data, over 12000 lines are drawn which heavily cross each other, meaningful results are hard to detect. Therefore this view supports showing only small subsets of data: If there are selections made from the data, not the entire data is displayed in this view, but the data containing (only) the first selection.

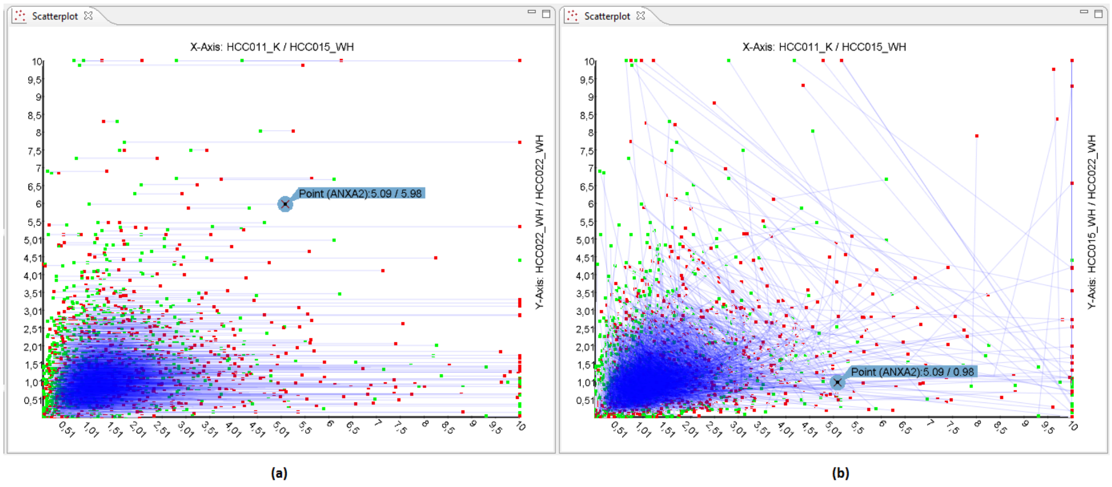

Scatterplot
X-Axis: HCC011_K / HCC015_WH
Point (ANXA2):5.09 / 5.98
Y-Axis: HCC022_WH / HCC022_WH
(a)
Scatterplot
X-Axis: HCC011_K / HCC015_WH
Point (ANXA2):5.09 / 0.98
Y-Axis: HCC015_WH / HCC022_WH
(b)

Figure 5.6: *Quad-vise plots in Caleydo. (a) The correlation between a pair of experiments an an additional experiment, resulting in lines parallel to the x-axis. (b) The correlation between 2 pairs of experiments.*

5.2 Scattterplot Matrix

We chose not to employ any dimension reduction methods since we wanted to provide a full view of the data *as it is*. Therefore we visualize the data as full-scale scatterplot matrix with the main-view of two specific experiments embedded. Correlations to other experiments are provided with the *linking and brushing* technique both with the scatterplots and with the other, already implemented views in the Caleydo framework.

General Visualisation

The scatterplot matrix is visualized without the redundancy in the upper left half and the diagonal of the matrix. The diagonal is used for providing a histogram of the correlated experiment. Left of the histogram name-labels are shown identifying the corresponding experiments with small arrows pointing to the rows and columns.

The main view is embedded in the space not occupied by the matrix. Since the view sizes are adjustable at will in Caleydo, the embedded main view will place itself any-where where its area is maximized (with the constrain of a low disparity between the x- and the y-axis). This can be seen in Figure 5.7, where the size of the matrix changes zoom-feature. In the worst case, the mainview is allowed to occupy only a quarter of the screen.

The matrix cells are rendered as 2-layered textures, one layer for the base data and one layer for the brushed data (see Figure 5.8). The texture resolutions depends on the number of experiments, number of genes and the selected samples-size (See Section 4.3 for more details). Selections made in the main view are propagated immediately to each matrix cell. This is the most costly operation in the scatterplot view, since $(selectedgenes) * (experiments)^2$ operations need to be performed to update the whole matrix. For our test data with only 14 experiments and 3 selections of 5000 genes, that is already over 3 millions operations to be performed. On low-end systems, that can take some seconds actually. Using the build in random sampling described above is recommended here. On the positive side, the redrawing of the textures only take place when a brushing operations is finished, so it does not really hinder the standard workflow.

Matrix Zoom

The more experiments are contained in the data, the less space is available for each individual matrix-cell. Recognizing meaningful information in the data without very strong correlation between experiments might be impossible. For this reasons, instead of having to bring every projections in the main view, we give the user a tool to get details on demand directly in the scatterplot matrix.

Figure 5.7 shows the matrix-zoom in action. When this feature is enabled, the matrix-cell under the mouse pointer (or the current cursor position) is scaled up by the factor nine, its vicinal cells by the factor of 4. We also do not allow distortion, not even in the non-focus area. The price for this is that we lose some space, but the big advantage we get is that we keep the feeling of the rows and columns, which would be lost taking approaches as suggested in Section 3.3.

5.3 Selections and Selection Manager

Caleydo and in this particular case the scatterplot view offer three kinds of picking or brushing methods. If used in one view, they are propagated to other views as well:

- **Mouse-over picking:** Hovering the mouse over a data-item will highlight it and can provide additional information. In the case of scatterplots it shows its name, its affinity to specific selections and its value in both directions.
- **Selections per mouse-click picking:** The left mouse button allows an item to be selected, the right mouse button deselects the item.
- **Selections via view-specific brushes:** Each view can provide a special brushing technique related to its kind of visualisation, parallel coordinates support for example angular brushing [Lex2008].

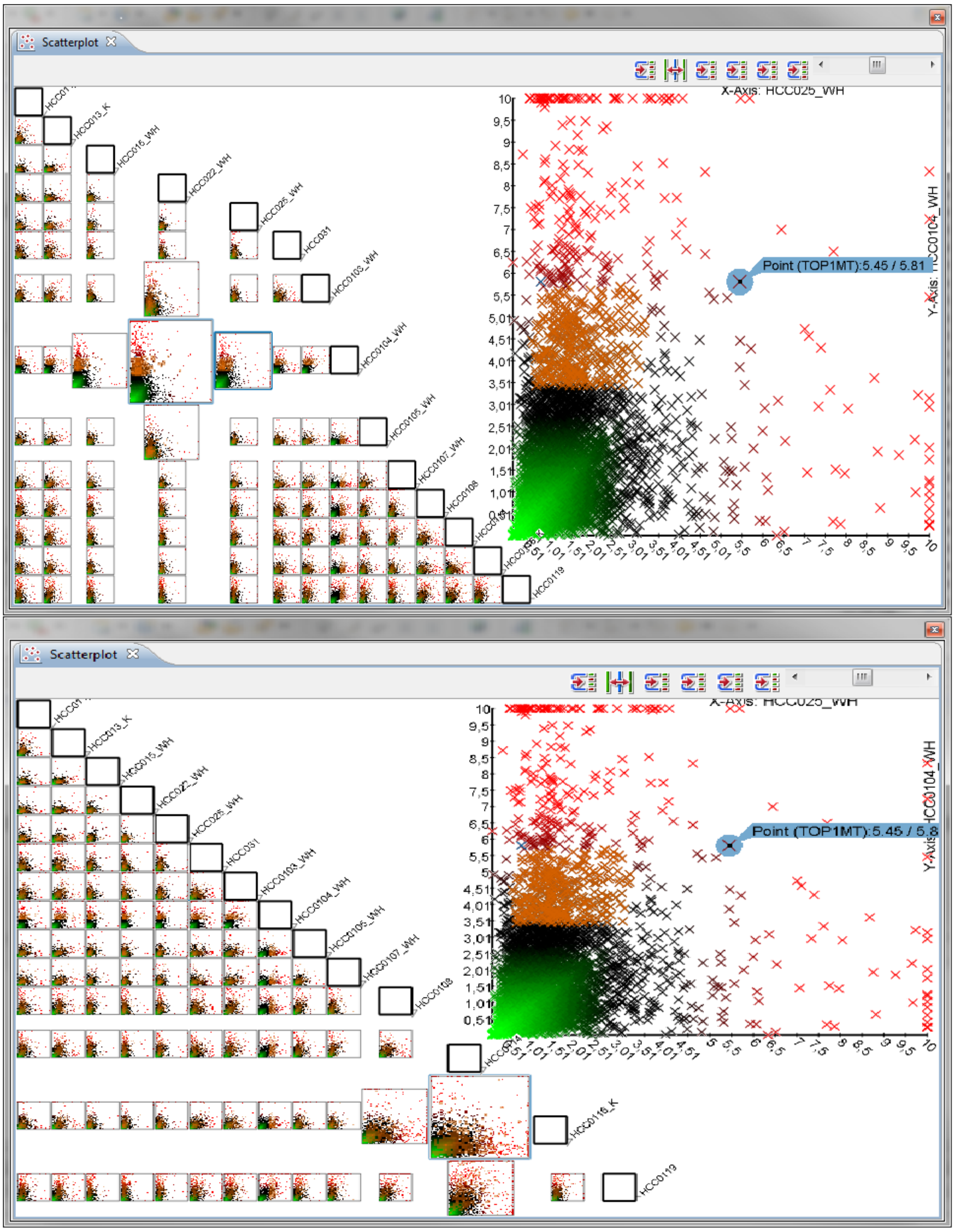


Figure 5.7: *Scatterplot Matrix Zoom.*

Brushing in the Scatterplot

Currently only one type of brush is implemented, the selection via rectangular window. Holding down the left mouse-button and dragging the mouse over data-points forms a green rectangle. As soon as the mouse-button is released, the brushed data inside the rectangle is highlighted and the selection is propagated to the framework, which updates the corresponding genes to other views. In the scatterplot matrix, selections are drawn with only half the resolution (thus providing a higher visibility of each selected data-point) compared to the rest of the data, since small selections may be hard to see there. See Figure 5.8. At this time, only binary brushing is supported.

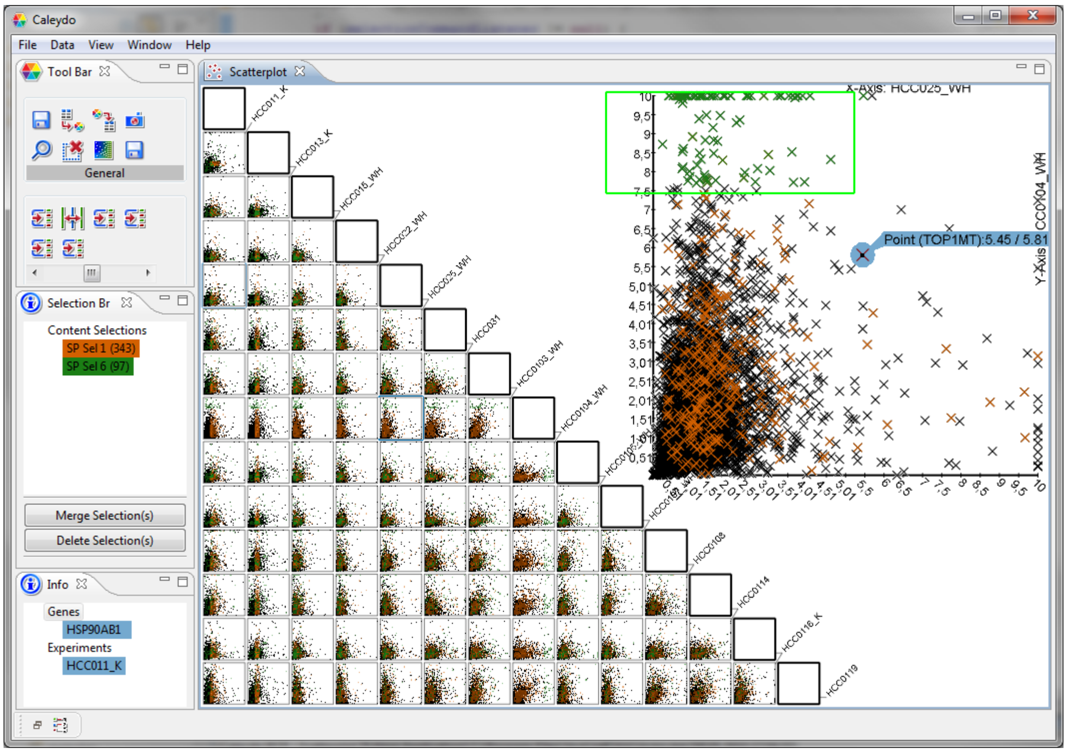


Figure 5.8: *Scatterplot Matrix Selection*

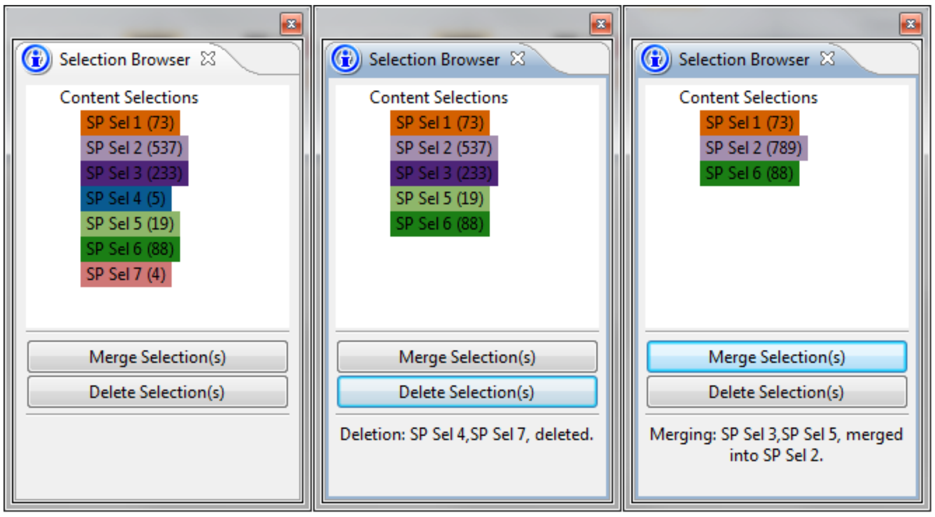
The Selection Browser

Since Caleydo supports multiple brushes, there is a need for a tool to administrate them. This is accomplished by the selection manager (see Figure 5.9).

Every time a brushing operation is performed, a new brush is built. The color of the brush is predetermined, but follows a scheme for optimal color-coding for maps[1]. Each new selection has a higher intern priority value then its predecessors, so they are rendered over them if two selections contain the same data. Every selection is also given a name upon creation, normally consisting of the abbreviation of the view and an incremental number.

In the selection manger all selections are shown with their names, brush-color and the num-

[1]http://colorbrewer2.org/


Selection Browser

Content Selections
SP Sel 1 (73)
SP Sel 2 (537)
SP Sel 3 (233)
SP Sel 4 (5)
SP Sel 5 (19)
SP Sel 6 (88)
SP Sel 7 (4)

Merge Selection(s)
Delete Selection(s)

Selection Browser

Content Selections
SP Sel 1 (73)
SP Sel 2 (537)
SP Sel 3 (233)
SP Sel 5 (19)
SP Sel 6 (88)

Merge Selection(s)
Delete Selection(s)

Deletion: SP Sel 4,SP Sel 7, deleted.

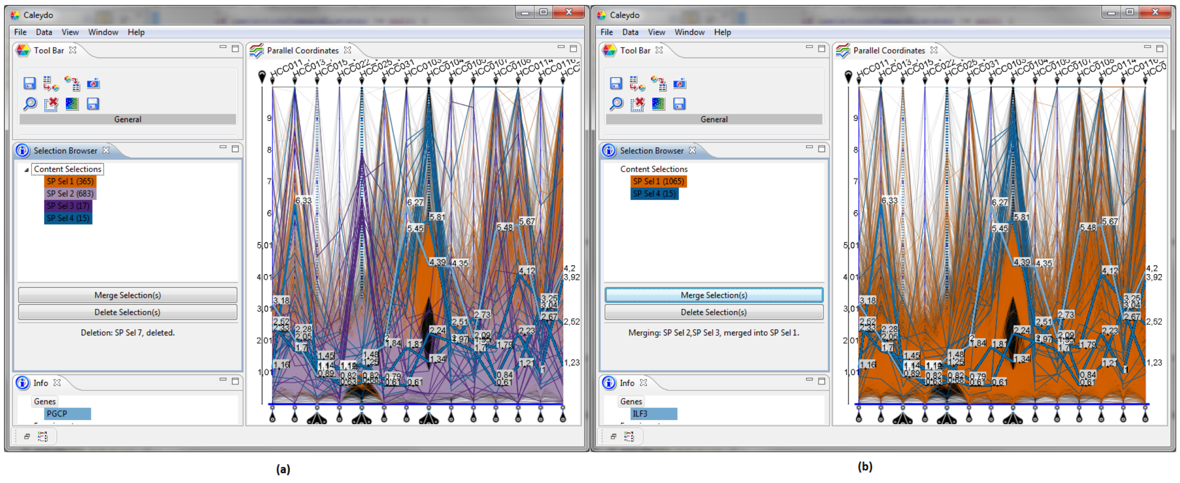
Selection Browser

Content Selections
SP Sel 1 (73)
SP Sel 2 (789)
SP Sel 6 (88)

Merge Selection(s)
Delete Selection(s)

Merging: SP Sel 3,SP Sel 5, merged into SP Sel 2.

Figure 5.9: *Selection Manager showing deletion and merging operations*



Deletion: SP Sel 7, deleted.

Merging: SP Sel 2,SP Sel 3, merged into SP Sel 1.

Figure 5.10: *Selection Manager in action with parallel coordinates.(a) 4 Selections (SP Sel 1-4) are available. (b) Selection SP Sel 2 and SP Sel 3 are merged into selection SP Sel 1*

ber of genes they contain. The selection browser supports operations like merging multiple selections into one and the deletions of one or more selections. Figure 5.10 shows the selection browser in action with the parallel coordinates view.

Chapter 6

Conclusions and Future Work

In this work we created a view for visualizing gene expression data based on scatterplots. One of the key aspects was to provide a smooth integration into the Caleydo framework to provide a consistent multiple coordinated view experience (see Figure 6.1).

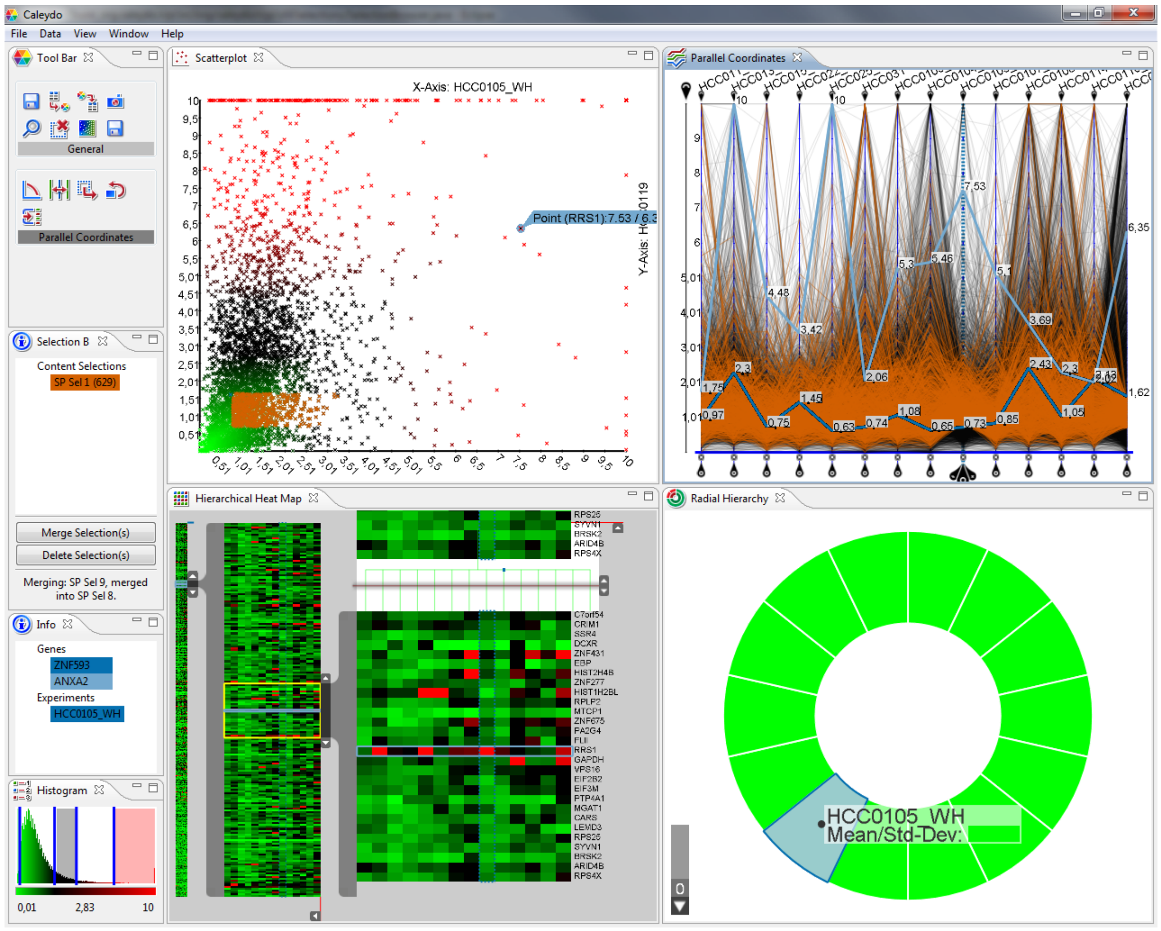


Figure 6.1: *Multiple Coordinated Views in Caleydo featuring scatterplot, heatmap, radial hierarchy and parallel coordinates, histogram, selection browser and info area.*

We have done this by providing a feature-rich main view to visualize the correlation between two experiments, with the help of quad-vise plots even four experiments. While the main view itself is very customizable in its visual appearance, some aspects are directly inherited from the framework itself, like the color-coding of the gene expressions or the appearance of the standard brushes. For viewing the whole n-dimensional data-set, we provided a high performance scatterplot matrix, which directly embeds the main-view of the currently selected experiments in the space which the redundancy of a scatterplot opens. Therefore, we get a *Focus plus Context* system where the scatterplot matrix serves as overview for the actual scatterplot.

For indicating correlations between more than four experiments and even across the framework, we implemented a *Linking and Brushing* method (see Chapter 3.4), which propagates the brushes taken in the main view to all matrix projections and other views as well. We have done this by enhancing the Caleydo framework to support multiple user-generated brushes. For a better management of those, we created a selection browser which can visualize, merge and delete brushes by user interaction.

Future Work

The currently implemented rectangular brush is easy to use and with the additional single point picking and the ability to merge brushes with the selection browser, also a very flexible one. Still, for arbitrarily shaped selections, there are faster and better methods like polygonal or lasso brushes. In the framework, this is a straightforward enhancement, only two methods need to be implemented per new brush: a method that draws the new brush and another evaluator method that performs an in/out test for the chosen brushing shape.

In this work, we chose to not rely on dimension reduction methods but rather present the whole data in form of a scatterplot matrix all the time, for reasons discussed in Chapter 4. Nevertheless, dimension reduction methods can reveal correlations which the current techniques may not be able to, so implementing these techniques might be a valuable addition. Selecting the dimensions on which the dimension reduction methods are performed normally happen in an extra window, we actually do not need to implement a special GUI for this, we can enhance our current system to use the scatterplot matrix as *dimension-selector*. Currently the matrix is treated as a selector of two (or four in the case of quad-vise-plots) dimensions, but we do not need to limit the number of projections, we can allow an arbitrary number by generalising this idea. From the user's point of view, he could select a random number of dimensions via shift-clicking on various matrix cells, where in the main-view a PCA or a similar projection method is then performed to visualize the correlation between all those selected dimensions.

While the current implementation runs sufficiently fast even on low end systems with typical gene expression data (See Chapter 4), the Caleydo framework is not limited to any kind of data. An excessive number of dimensions are able to reduce the responsiveness dramatically due the square order of the program runtime when a scatterplot-matrix is involved. Since we followed the *Layered Rendering* approach we discussed in Section 2.1.5 and Chapter 4, the next logical step would be to implement Piringers [Piringer2009] *Early Thread Termination* to make full use of modern, multi-core CPUs.

Although Berhard Schlegl already compared the Caleydo framework with similar frameworks in his master thesis [Schlegl2009], his focus was on the visual analytics aspects and scatterplot played only a small role in his evaluation. So the next logical step for this work is to investigate the implementation details and techniques of scatterplots in those frameworks in comparison to our work.

Appendix A

Keyboard shortcuts

Toolbar Action	Key	Description
Toggle Point Type	p	Change the point-primitives in the main view: (box, point, disk, circle, cross)
Toggle Matrix View/Main View	m	Switch between main view only and embedded view
Toggle Matrix Zoom Mode	o	Mouseover in the Matrix zooms out the selected matrix cell and its neighbours, simulating a 'discret fisheye'
Toggle Main View Zoom Mode	z	Enables a scalable and moveable zoom-window in the main view.The zoom-window position is reset by disabling this mode
Coloured Scatterplots	c	Switches between black-only or color-coded scatterpoints
Enable/Disable Quad-vise Mode	b	2 Experiments are rendered each axis, connected with a line to visualize the correlation between 4 genes
Point Size Slider	-	The slider widget lets you scale the scatter-points within a predetermined range

Table A.1: *Keyboard shortcuts and toolbar actions for the Caleydo scatterplot view*

Additional Keyboard commands:

- **Arrow-Keys:** In the main view only, you can directly navigate through the experiments. In the scattermatrix view, you can move the Mouse-over selection.
- **CTRL+Arrow-Keys:** In the quad-vise plot mode, you can navigate with the second pair of axis through the matrix.
- **ENTER:** The in the mouse-over selection highlighted matrix-element will be rendered in the main view.

List of Figures

List of Tables

Listings

Bibliography

[Alberts2002] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Molecular Biology of the Cell, Fourth Edition*. Garland Science, Taylor & Francis Group, New Yorks, fourth edition, **2002**. ISBN 0815332181.

[Andrienko2007] Gennady Andrienko and Natalia Andrienko. *Coordinated multiple views: a critical view*. In *CMV '07: Proceedings of the Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization*, pp. 72–74. IEEE Computer Society, Washington, DC, USA, **2007**. ISBN 0-7695-2903-8.

[Aris2007] Aleks Aris and Ben Shneiderman. *Designing semantic substrates for visual network exploration. Information Visualization*, volume 6(4):pp. 281–300, **2007**. ISSN 1473-8716.

[Asimov1994] Daniel Asimov and Andreas Buja. *The grand tour via geodesic interpolation of 2-frames*. In *in Visual Data Exploration and Analysis, Symposium on Electronic Imaging Science and Technology, IS&T/SPIE (Soc. for Imaging Sci. and Technology/Internat. Soc. for Optical Engineering*. **1994**.

[Bachthaler2008] S. Bachthaler and D. Weiskopf. *Continuous scatterplots. Visualization and Computer Graphics, IEEE Transactions on*, volume 14(6):pp. 1428 –1435, **nov.-dec. 2008**. ISSN 1077-2626.

[Bajaj1997] Chandrajit L. Bajaj, Valerio Pascucci, and Daniel R. Schikore. *The contour spectrum*. In *VIS '97: Proceedings of the 8th conference on Visualization '97*, pp. 167–ff. IEEE Computer Society Press, Los Alamitos, CA, USA, **1997**. ISBN 1-58113-011-2.

[Baldonado2000] Michelle Q. Wang Baldonado, Allison Woodruff, and Allan Kuchinsky. *Guidelines for using multiple views in information visualization*. In *AVI '00: Proceedings on Advanced visual interfaces*, pp. 110–119. ACM Press, New York, NY, USA, **2000**.

[Becker1987] Richard A. Becker and William S. Cleveland. *Brushing scatterplots. Technometrics*, volume 29(2):pp. 127–142, **1987**.

[Bertin1974] Jacques Bertin and Georg Jensch. *Graphische Semiologie: Diagramme, Netze, Karten*. de Gruyter, Berlin, first published 1967 in french, german edition, **1974**.

[Bier1993] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. *Toolglass and magic lenses: the see-through interface*. In *SIGGRAPH '93: Proceedings of the 20th annual conference on*

Computer graphics and interactive techniques, pp. 73–80. ACM Press, New York, NY, USA, **1993**.

[Buja1991] A. Buja, J.A. McDonald, J. Michalak, and W. Stuetzle. *Interactive data visualization using focusing and linking.* In *Visualization, 1991. Visualization '91, Proceedings., IEEE Conference on*, pp. 156 –163, 419. **oct 1991**.

[Card1999] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman, editors. *Readings in information visualization: using vision to think.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, **1999**. ISBN 1-55860-533-9.

[Carr1987] D. B. Carr, R. J. Littlefield, W. L. Nicholson, and J. S. Littlefield. *Scatterplot matrix techniques for large n. Journal of the American Statistical Association*, volume 82(398):pp. 424–436, **1987**. ISSN 01621459.

[Carr2006] H. Carr, B. Duffy, and B. Denby. *On histograms and isosurface statistics. Visualization and Computer Graphics, IEEE Transactions on*, volume 12(5):pp. 1259 –1266, **sept.-oct. 2006**. ISSN 1077-2626.

[Chen2004] Hong Chen. *Compound brushing explained. Information Visualization*, volume 3(2):pp. 96–108, **2004**. ISSN 1473-8716.

[Chi1997] E.H.-H. Chi, P. Barry, J. Riedl, and J. Konstan. *A spreadsheet approach to information visualization.* In *Information Visualization, 1997. Proceedings., IEEE Symposium on*, pp. 17 –24. **oct. 1997**.

[Cleveland1988] William C. Cleveland and Marylyn E. McGill. *Dynamic Graphics for Statistics.* CRC Press, Inc., Boca Raton, FL, USA, **1988**. ISBN 053409144X.

[Cleveland1984] William S. Cleveland and Robert McGill. *The many faces of a scatterplot. Journal of the American Statistical Association*, volume 79(388):pp. 807–822, **1984**. ISSN 01621459.

[Collins2007] Christopher Collins and Sheelagh Carpendale. *Vislink: Revealing relationships amongst visualizations. IEEE Transactions on Visualization and Computer Graphics*, volume 13(6):pp. 1192–1199, **2007**. ISSN 1077-2626.

[Convertino2003] G. Convertino, J. Chen, B. Yost, Y.-S. Ryu, and C. North. *Exploring context switching and cognition in dual-view coordinated visualizations.* In *Coordinated and Multiple Views in Exploratory Visualization, 2003. Proceedings. International Conference on*, pp. 55 – 62. **july 2003**.

[Asimov1985] D.Asimov. *The grand tour: A tool for viewing multidimensional data. SIAM*, volume 6:pp. 128–143, **1985**.

[Dix2002] Alan Dix and Geoff Ellis. *By chance: enhancing interaction with large data sets through statistical sampling. Proceedings of Advanced Visual Interfaces - AVI2002*, pp. 167–176, **2002**.

[Doleisch2002] Helmut Doleisch and Helwig Hauser. *Smooth brushing for fo-*

cus+context visualization of simulation data in 3d. In *WSCG*, pp. 147–154. **2002**.

[Dupont2003] William D. Dupont and W. Dale Plummer Jr. *Density Distribution Sunflower Plots*, volume 8. **1 2003**.

[Elmqvist2008] N. Elmqvist, P. Dragicevic, and J.-D. Fekete. *Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. Visualization and Computer Graphics, IEEE Transactions on*, volume 14(6):pp. 1539 –1148, **nov.-dec. 2008**. ISSN 1077-2626.

[Friedman1974] J.H. Friedman and J.W. Tukey. *A projection pursuit algorithm for exploratory data analysis. Computers, IEEE Transactions on*, volume C-23(9):pp. 881 – 890, **sept. 1974**. ISSN 0018-9340.

[Furnas1986] George W. Furnas. *Generalized fisheye views.* In *CHI '86: Proceedings on Human factors in computing systems*, pp. 16–23. ACM Press, New York, NY, USA, **1986**.

[Gamma1995] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, **1995**. ISBN 0-201-63361-2.

[Grubbs1969] F. E Grubbs. *Procedures for detecting outlying observations in samples.* 11. Technometrics, **1969**.

[Hauser2002] Helwig Hauser, Florian Ledermann, and Helmut Doleisch. *Angular brushing of extended parallel coordinates.* In *INFOVIS '02: Proceedings on Information Visualization*, pp. 127–130. IEEE Computer Society, Washington, DC, USA, **2002**.

[Hibbs2005] M. A. Hibbs, N. C. Dirksen, K. Li, and O. G. Troyanskaya. *Visualization methods for statistical analysis of microarray clusters. BMC Bioinformatics*, volume 6:p. 115, **2005**. ISSN 1471-2105.

[Inselberg1985] Alfred Inselberg. *The plane with parallel coordinates. The Visual Computer*, volume 1(4):pp. 69–91, **1985**.

[Kalkusch2006] Michael Kalkusch and Dieter Schmalstieg. *Extending the scene graph with a dataflow visualization system.* In *VRST 2006: Proceedings on Virtual reality software and technology*, pp. 252–260. ACM, New York, NY, USA, **2006**.

[Keim2002] Daniel A. Keim. *Information visualization and visual data mining. IEEE Transactions on Visualization and Computer Graphics*, volume 8(1):pp. 1–8, **2002**. ISSN 1077-2626.

[Keim2005] Daniel A. Keim. *Scaling visual analytics to very large data sets.* Presentation at the Workshop on Visual Analytics, **June 2005**.

[Koren2004] Y. Koren and L. Carmel. *Robust linear dimensionality reduction. Visualization and Computer Graphics, IEEE Transactions on*, volume 10(4):pp. 459 –470, **july-aug. 2004**. ISSN 1077-2626.

[Kosara2003] Robert Kosara, Helwig Hauser, and Donna L. Gresh. *An interaction view on information visualization.* In *State-of-the-Art Proceedings of EUROGRAPHICS 2003 (EG 2003)*, pp. 123–137. **2003**.

[Kosara2002] Robert Kosara, Silvia Miksch, Helwig Hauser, Johann Schrammel, Verena Giller, and Manfred Tscheligi. *Useful properties of semantic depth of field for better f+c visualization.* In *VISSYM '02: Proceedings of the symposium on Data Visualisation 2002*, pp. 205–210. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, **2002**. ISBN 1-58113-536-X.

[Lex2008] Alexander Lex. *Exploration of Gene Expression Data in a Visually Linked Environment.* Master's thesis, Graz University of Technology, **2008**.

[Lex2010] Alexander Lex, Marc Streit, Ernst Kruijff, and Dieter Schmalstieg. *Caleydo: Design and evaluation of a visual analysis framework for gene expression data in its biological context.* In *To appear in: Proceedings of 2010 IEEE Pacific Visualization Symposium, Taipeh, Taiwan.* **2010**.

[Li2009] Jing Li, Jarke J. van Wijk, and Jean-Bernard Martens. *Evaluation of symbol contrast in scatterplots.* In *PACIFICVIS '09: Proceedings of the 2009 IEEE Pacific Visualization Symposium*, pp. 97–104. IEEE Computer Society, Washington, DC, USA, **2009**. ISBN 978-1-4244-4404-5.

[Mackinlay1991] Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. *The perspective wall: detail and context smoothly integrated.* In *CHI 1991: Proceedings on Human factors in computing systems*, pp. 173–176. ACM Press, New York, NY, USA, **1991**.

[Martin1995] Allen R. Martin and Matthew O. Ward. *High dimensional brushing for interactive exploration of multivariate data.* In *VIS '95: Proceedings on Visualization '95*, p. 271. IEEE Computer Society, Washington, DC, USA, **1995**.

[Moore1965] Gordon E. Moore. *Cramming more components onto integrated circuits. Electronics*, volume 38(8), **April 1965**.

[Murrell2006] Paul Murrell. *Graphics of large datasets: Visualizing a million. Journal of Statistical Software*, volume 17(b01), **2006**.

[Pearson1901] K. Pearson. *On lines and planes of closest fit to systems of points in space. The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, volume 2:pp. 559–572, **1901**.

[Peng2004] Wei Peng, Matthew O. Ward, and Elke A. Rundensteiner. *Clutter reduction in multi-dimensional data visualization using dimension reordering.* In *INFOVIS '04: Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'04)*, pp. 89–96. IEEE Computer Society, Washington, DC, USA, **2004**. ISBN 0-7803-8779-3.

[Piringer2009] H. Piringer, C. Tominski, P. Muigg, and W. Berger. *A multi-threading* *architecture to support interactive visual exploration. Visualization and Computer Graphics, IEEE Transactions on*, volume 15(6):pp. 1113 – 1120, **nov.-dec. 2009**. ISSN 1077-2626.

[Plumlee2006] Matthew D. Plumlee and Colin Ware. *Zooming versus multiple window interfaces: Cognitive costs of visual comparisons. ACM Trans. Comput.-Hum. Interact.*, volume 13(2):pp. 179–209, **2006**. ISSN 1073-0516.

[Rabenhorst1994] David A. Rabenhorst. *Interactive exploration of multidimensional data.* In *Proceedings of the SPIE Symposium on Electronic Imaging*, pp. 277–286. **1994**.

[Rezaeian2009] Mohsen Rezaeian. *Getting to know the scatter plot. Middle East Journal of Age and Ageing*, volume 6 - Issue 2, **2009**.

[Roberts2007a] Jonathan C. Roberts. *State of the art: Coordinated & multiple views in exploratory visualization.* In *CMV '07: Proceedings of the Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization*, pp. 61–71. IEEE Computer Society, Washington, DC, USA, **2007**. ISBN 0-7695-2903-8.

[Robertson1993] George G. Robertson and Jock D. Mackinlay. *The document lens.* In *UIST '93: Proceedings of the 6th annual ACM symposium on User interface software and technology*, pp. 101–108. ACM, New York, NY, USA, **1993**. ISBN 0-89791-628-X.

[Sarkar1994] Manojit Sarkar and Marc H. Brown. *Graphical fisheye views. Commun. ACM*, volume 37(12):pp. 73–83, **1994**. ISSN 0001-0782.

[Sarkar1993] Manojit Sarkar, Scott S. Snibbe, Oren J. Tversky, and Steven P. Reiss. *Stretching the rubber sheet: a metaphor for viewing large layouts on small screens.* In *Proceedings of the 6th annual ACM symposium on User interface software and technology*, pp. 81–91. ACM, Atlanta, Georgia, United States, **1993**. ISBN 0-89791-628-X.

[Schall1995] Matthew Schall. *Diamond and ice: Visual exploratory data analysis tools. Perspective*, volume 18(2):pp. 15–24, **Nov 1995**.

[Schlegl2009] Bernhard Schlegl. *Visual Analytics for Gene Expression Data.* Master's thesis, Graz University of Technology, **2009**.

[Schmidt2000] Douglas Schmidt, Michael Stal, Hans Rohnert, and Frank Buschmann. *Pattern-oriented software architecture volume 2: Patterns for concurrent and networked objects.* **September 2000**.

[Seo2002] Jinwook Seo and Ben Shneiderman. *Interactively exploring hierarchical clustering results. Computer*, volume 35(7):pp. 80–86, **2002**. ISSN 0018-9162.

[Shneiderman1994] B. Shneiderman. *Dynamic queries for visual information seeking. Software, IEEE*, volume 11(6):pp. 70 –77, **nov 1994**. ISSN 0740-7459.

[Shneiderman1996] Ben Shneiderman. *The eyes have it: A task by data type taxonomy* *for information visualizations*. In *VL '96: Proceedings on Visual Languages*. IEEE Computer Society, **1996**. ISBN 081867508X.

[Shneiderman2006] Ben Shneiderman and Aleks Aris. *Network visualization by semantic substrates. IEEE Transactions on Visualization and Computer Graphics*, volume 12(5):pp. 733–740, **2006**. ISSN 1077-2626.

[Streit2007] Marc Streit. *Metabolic Pathway Visualization Using Gene-Expression Data*. Master's thesis, Graz University of Technology, **2007**.

[Streit2008] Marc Streit, Michael Kalkusch, Karl Kashofer, and Dieter Schmalstieg. *Navigation and exploration of interconnected pathways. Computer Graphics Forum (EuroVis 2008)*, volume 27(3):pp. 951–958(8), **May 2008**.

[Streit2009a] Marc Streit, Alexander Lex, Michael Kalkusch, Kurt Zatloukal, and Dieter Schmalstieg. *Caleydo: Connecting pathways with gene expression. Bioinformatics*, **2009**.

[Sutter2005] Herb Sutter. *The free lunch is over: A fundamental turn toward concurrency in software. Dr. Dobbs Journal*, volume 30(3):pp. 202–210, **2005**.

[Tague2004] Nancy R. Tague. *The Quality Toolbox*. ASQ Quality Press, second edition edition, **2004**.

[Thomas2005] James J. Thomas and Kristin A. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Ctr, **2005**. ISBN 0769523234.

[Tufte1983] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, Coneeticut, second edition, **1983 (Reprint 2001)**.

[Voigt2002] Robert Voigt. *An Extended Scatterplot Matrix and Case Studies in Information Visualization. Published as Diplomarbeit*. Master's thesis, **October 2002**.

[Ward1994] Matthew O. Ward. *Xmdvtool: integrating multiple methods for visualizing multivariate data*. In *VIS '94: Proceedings of the conference on Visualization '94*, pp. 326–333. IEEE Computer Society Press, Los Alamitos, CA, USA, **1994**. ISBN 0-7803-2521-4.

[Wattenberg2005] M. Wattenberg. *A note on space-filling visualizations and space-filling curves*. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, pp. 181 – 186. **23-25 2005**.

[Wegman2002] J. Wegman and Jeffrey L. Solka. *On some mathematics for visualizing high dimensional data*, **2002**.

[Yang2007] J. Yang, D. Hubball, M.O. Ward, E.A. Rundensteiner, and W. Ribarsky. *Value and relation display: Interactive visual exploration of large data sets with hundreds of dimensions. Visualization and Computer Graphics, IEEE Transactions on*, volume 13(3):pp. 494 –507,

may-june 2007. ISSN 1077-2626.