Visualizing the Effects of Logically Combined Filters

Thomas Geymayer*

Institute for Computer Graphics and Vision
Graz University of Technology
Graz / Austria

Abstract

Filtering data is an essential process in a drill-down analysis of large data sets. Filtering can be necessary for several reasons. The main objective for filters is to uncover the relevant subsets of a dataset. Another, equally relevant goal is to reduce a dataset to dimensions to which either visualization or algorithmic analysis techniques scale. However, with multiple filters applied and possibly even logically combined, it becomes difficult for users to judge the effects of a filter chain. In this paper we present a simple, yet effective way to interactively visualize a sequence of filters and logical combinations of these. Such a visualized *filter-pipeline* allows analysts to easily judge the effect of every single filter and also their combination on the data set under investigation and therefore, leads to a faster and more efficient workflow.

We also present an implementation of the proposed technique in an information visualization framework for the life sciences. The technique, however, could be employed in many other information visualization contexts as well.

Keywords: filter-pipeline, brushing, logical operations, interactive, data analysis, compound filter

1 Introduction

Visualizing large amounts of data has been one of the grand challenges of information visualization for over a decade now. With ever more data being produced, the ability to efficiently extract knowledge out of data becomes more important. There are several ways to analyze large quantities of data. Examples are aggregation or drill-down techniques, focus and context methods, and so on. In the sense of visual analytics [12, 7], visualization is combined with computational methods, such as machine learning or statistics. However, in many cases, raw data has several undesired attributes: parts of it can be redundant, noisy or irrelevant for a given task. Also, most methods – either computational or visual – do not scale arbitrarily. Fortunately, there is a simple and yet effective method to reduce the data to a manageable size: filtering. Filtering allows parts of the data to be removed, based on a given criteria. A filter can be defined visually or textually as a processing rule. Filters can be based on fairly simple concepts, such as thresholds, or on more complex processes, such as a statistical evaluation of significance. Related concepts are dynamic querying [1] (selecting only a desired subset of a data set instead of removing undesired parts) and to some extent, also brushing (highlighting a subset of a data set).

It is common to use a combination of filters to continually refine the analysis result. In many cases, such combinations are equivalent to logical operations [10, 5]. While a logical AND is the most commonly used, other operations such as OR, XOR and NOT are feasible as well.

While the reduced data set itself becomes more manageable, the overall filtering process and the individual effects of filters on the data set becomes increasingly obfuscated. To alleviate this, methods to visualize the combination of applied filters have been developed. Hong Chen [4] for example visualizes filters and other parameters of the visualization pipeline in a graph. However, to our knowledge, there has not been any technique that conveys not only the sequence of filters or brushes, but also the effects on the data size. Inspired by Minard's work, the famous *Carte Figurative des pertes successives en hommes de l'arme franaise dans la campagne de Russie 1812-1813* [13], which shows the continuous reduction of men in Napoleon's army during his Russian campaign, we have developed a visualization technique showing the effects of individual filters on a data set.

Our primary contribution is an interactive visualization technique for the effects of multiple filters, including the effects of logical operators applied to combinations of filters. This visualization technique enables users to understand the effects of individual and combined filters. A secondary contribution is a general and detailed analysis of requirements for visualizing multiple filters. Having these requirements at hand, we demonstrate how the proposed technique satisfies each of the specified requirements.

2 Related Work

Much of how we interact with large quantities of data in visualization has its roots in the 1980s and early 1990s. Becker proposed some basic principles for dynamic data analysis [2], like linking & brushing – a technique that is commonly used until today. In 1992 Ahlberg *et al.* [1]

*tomgey@gmail.com

conducted an experiment with dynamic queries performed on a database with different combinations of graphical and textual input and output, respectively. As the different parameters used for the dynamic queries have the effect of refining the data set, this work is an early example of dynamically adapted filters.

In 1995 Martin and Ward propose an improvement of the XmdvTool which contains methods to combine multiple brush operations with different logical operations [10].

A recent example of an approach of filtering high dimensional data can be found in [14] where cross-filtering across multiple views is presented.

These works lay the foundation for modern visualization systems which widely employ combined filters or brushes to refine data sets or selections. However, only very few systems visualize these combinations in an explicit way. One notable exception is the work by Hong Chen [4] where node-link diagrams are used to visualize operations like a brush or selections. He also employs combined nodes which visualize logical or analytical operations. However, while individual operations are visualized, the effects of the operations on the data are not.

3 Problem Analysis

Users often find it hard to remember the steps conducted to get a specific result [14]. To support analysts and reduce the required cognitive load, we believe that an explicit representation of the filter sequence helps to understand the interdependencies between and consequences of filters. We elicited the following main requirements for such a filter-pipeline meta-visualization technique:

1. **Show Sequence**
   As filters are typically applied sequentially, it is essential to show the filters in the sequence they were applied.

2. **Show Consequences**
   To allow a user to judge how much data is removed by a filter, a filter visualization needs to show how much elements a filter reduces.

3. **Show and Create Compositions**
   A simple sequence of filters is equal to logical AND operations (*i.e.*, show all elements which are not removed by filter X and filter Y). Other logical operations such as OR and XOR cannot be visualized as easily. It is essential, that such compositions are adequately represented in a dedicated filter visualization technique.
   Additionally to the visualization of composed filters, it should also be possible to create composed filters based on pre-existing filters.

Aside from these main requirements for a filter visualization technique, there are several other requirements which provide added benefit to users:

4. **Modify Filters**
   An interactive filter visualization technique should enable a user to modify a filter (*i.e.*, change its parameters), to remove a filter and to move a filter to another position in the sequence of filters.

5. **Hide Filters**
   In some cases it may be desirable to hide filters. A common example is an initial filter that removes noisy data. If such a filter removes a lot of data items, the consequences of subsequently applied, fine-grained filters become hard to perceive, due to the small change relative to the initial filter. One solution would be to use logarithmic scales for the amount of data removed. However, as log scales are not intuitive, we believe that the ability to hide filters is superior.

6. **Show Filter Efficiency**
   When a filter is visualized in relation to a previous filter, it is impossible to judge its effect on the global data set, since only the effects on the already filtered data set is shown. To make a user understand the consequences and the efficiency of a filter better, an effective filter visualization technique should also enable a user to see how much data a filter would remove from the complete data set.

In the following chapter, we will describe how we address each of these requirements and challenges to create a simple and yet effective filter visualization technique.

4 Method

Similar to the visualization of the reduction of men in Napoleon's army [13], we render each filter as a quadrangle where the left side represents the input and the right side the output of the filter (see Figure 1). The size is chosen in way that the largest visible filter always fills the available height, and the length is equally distributed over all visible filters. The height of the left edge of the filter encodes the elements going in, while the height of the right edge encodes the elements going out of a filter. Consequently, the difference in height (which is known to be the most powerful visual variable [3]) as well as the slope of the top edge allow to easily judge the effect of the filter. To convey a sense of absolute numbers, we also chose to show the number of current elements and the number of elements each filter removes from its input.

4.1 Basic Sequence of Filters

We show a sequence of filters as an equivalent to the logical AND operation, which simply concatenates one filter after each other, such that the output of a filter is passed to the next filter as input. This simple, yet effective method satisfies Requirements 1 and 2.

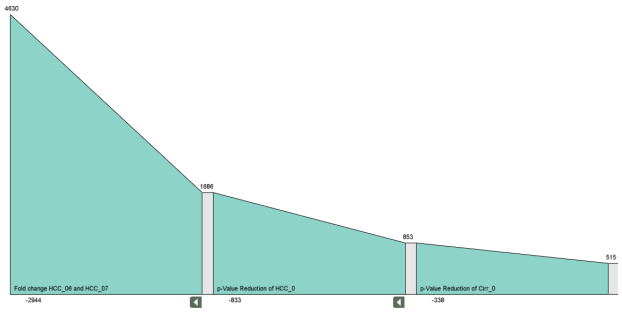

Figure 1: Sequence of filters. The result is equivalent to a logical AND operation of the filters.

4.2 Compound Filters

For more advanced filter-pipelines, combinations of multiple filters into a single filter can be necessary. We provide the possibility to create meta-filters where all involved (sub)-filters are combined in one filter in the sequence of top-level filters. This can be achieved by dragging an existing filter and dropping it onto another already created filter. The whole meta-filter's input data is passed to each sub-filter. The output is then calculated based on the desired logical operations – most commonly an OR.

In order to visualize this combination, we experimented with two different approaches. One is to stack every involved sub-filter on top of each other and embed this stacking into the overall meta-filter. Alternatively, we embed all sub-filters in the top-level meta filter without overlaps. In the following, we will briefly discuss the advantages and disadvantages of each method.

Stacked Sub-Filters

As each sub-filter of the combined filter receives the same input, it is intuitive to render all sub-filters on top of each other at the same location (see Figure 2). The filters are sorted from top to bottom, where the topmost sub-filter (in our example rendered in a light purple color) is the least effective one (the one that removes the least elements from the input data), thus guaranteeing that no filter is completely occluded. The height on the left side of each filter is the total input of the compound filter and the height on the right side of each filter represents its individual output. Additionally, in the background, the union of all filters is rendered, visualizing the total influence of the combined filters on the filter-pipeline. In our example, this is the largest filter with the light yellow background.

It is also of interest, to know which part of the input that passes all filters, which is the intersection of all individual sub-filter outputs (the result of an AND operation). We visualize this by adding another quadrangle on top. According to the characteristics of set intersection this will always be the smallest quadrangle (in Figure 2 it is rendered in green). This information can also be confusing to the user and misinterpreted as an additional filter. Thus, we only show it if the user moves the mouse over the filter. This allows him to detect inefficient filters, *i.e.*, filters that only contribute few or even no elements, apart from the elements also contributed by the other filters.

A problem with this approach is that it becomes cluttered easily. We found that for as little as three filters, especially if they are similar in terms of their efficiency, it is not easy to distinguish individual filters. Furthermore, it discontinues the flow of the data through the filter-pipeline, as all sub-filters have dead ends on their right sides without an equivalent at the left side of the next filter. Consequently, we devised an alternative method which addresses both issues.

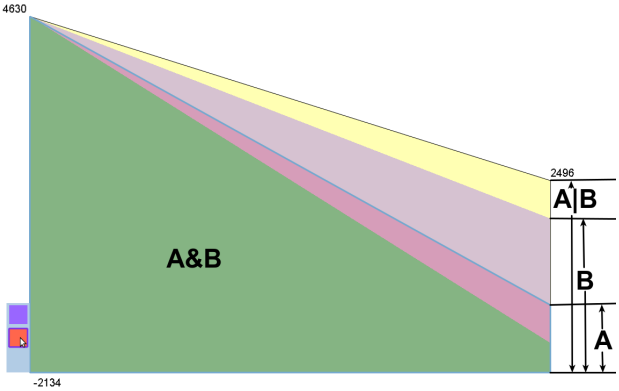

Figure 2: Two filters combined with logical OR, both sub-filters stacked over each other. The largest filter with the light yellow background is the resulting filter (A|B), the two purple filters are the combined sub-filters (just A or just B) and the green filter on top of all filters represents the intersection of the elements filtered by both sub-filters (A&B).

Separate Sub-Filters

As the sub-filters in a compound filter operate in parallel (contrary to the sequence of filters on the top-level), we considered to also express this property in the visualization technique. Consequently, we render the sub-filters at a smaller scale in parallel inside the resulting compound filter. To provide a continuous flow of the data through the filter-pipeline, we connect the input of the compound filter to each sub-filter using curved shapes (see Figure 3). The surfaces use the same color as the respective sub-filters, with transparency increased to allow a user to see the overlapping regions. Inspired by Kosara *et al.*'s work on categorical data visualization [8], we then calculate all possible intersections between the contributed elements of every sub-filter. Consequently, for a composition of two filters, if the underlying operation is an OR, there are two categories of elements. Those which are contributed by only one sub-filter, and those that are contributed by both. We render each set with a trapezoid using the same color as we did for the incoming surfaces (see Figure 3). To make the relative size of the set intersections more obvious, we use the space right of the filter, to show the set sizes in de-

tail. Moving the mouse over an intersection, shows which filters are intersected for this sub-set.

With the technique of using separate sub-filters embedded in a meta-filter, we have successfully addressed Requirement 3.

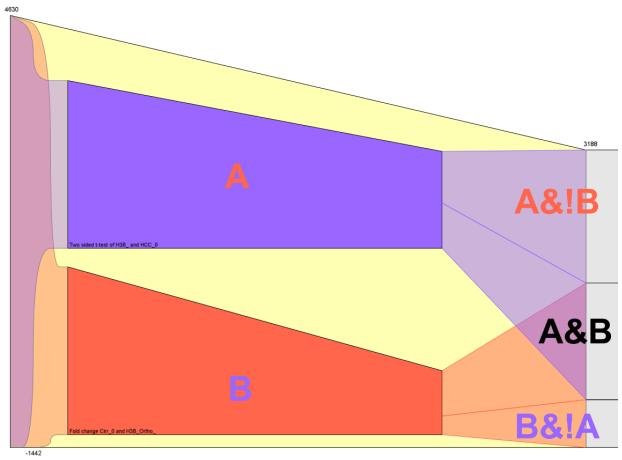

Figure 3: Two filters (labeled with A and B) combined with logical OR, visualized in parallel. The large filter in the background, rendered in light yellow, is the resulting filter. The left edge of the resulting filter is connected with the sub-filters using curved surface. On the right side, all intersections between the elements that are passed to at least one of the sub-filters are visualized – elements contributed only by filter A (A&!B), elements contributed only by filter B (B&!A), and elements contributed by both, filter A and B (A&B).

4.3 Modifying Filters

The described filter visualization lends itself to allow interaction with the filters themselves. As discussed in Requirement 4, the essential operations are: modify, remove and move. We provide intuitive access to those features, for example by drag and drop for moving filters, or by double clicking on a filter for modifying it.

4.4 Hiding Filters

We have already discussed the issue of combinations of strong filters that initially remove large portions of the data, and more sensitive, refining filters that remove only smaller parts (see Section 3, Requirement 5). Another issue, aside from the inability to perceive the effects of filters removing only a view elements, is the fact, that composed meta-filters containing several sub-filters are hard to see because of the tiny amount of space available. These problems are illustrated in Figure 4. As a solution to this problem, we provide the possibility to hide a number of filters at the front of the filter-pipeline. This way, the remaining filters can be scaled up to the whole height, which makes their subtle effects on the filter-pipeline, as well as the embedded filters visible again. The effect is shown in Figure 4.

Below each filter, there is a button that enables the analyst to hide all filters from the front up to the according filter. If at least one filter is hidden, we show a button on the left margin to again display the hidden filters.

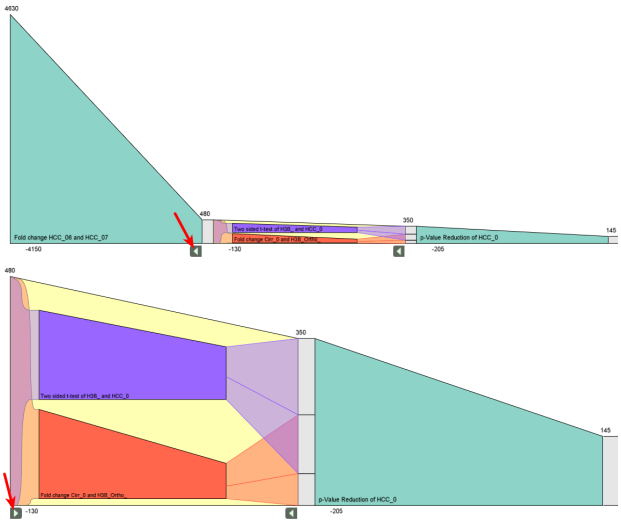

Figure 4: Hiding filters: By pressing the arrow button below the filter, the filter and all filters left of it are hidden. The example shown reduces the visible pipeline to only two filters which are scaled to the whole available height, as depicted in the lower image. Notice that the relative changes and the composition of the compound filter are much better visible when compared to the upper figure. By clicking on the button on the left border, the hidden filter can be made visible again.

4.5 Show Filter Efficiency

As every filter in the pipeline gets the output of its preceding filter, the amount of elements filtered is smaller (in most cases) than if it were applied on the whole input data set. However, as discussed in Section 3, Requirement 6, it can be useful to get an idea on how the filter would behave if it were applied to the whole data set. This, for example, is desirable when the data is filtered to meet a pre-condition for a feasible runtime of a given algorithm. In such a case, a user can apply different filters at the same time, and judge, whether he could meet the requirements with for example only one of the filters.

We address this challenge by overlaying an transparent version of the filter, showing its size as if it was the only filter in the pipeline, on mouse over. This is shown in Figure 5.

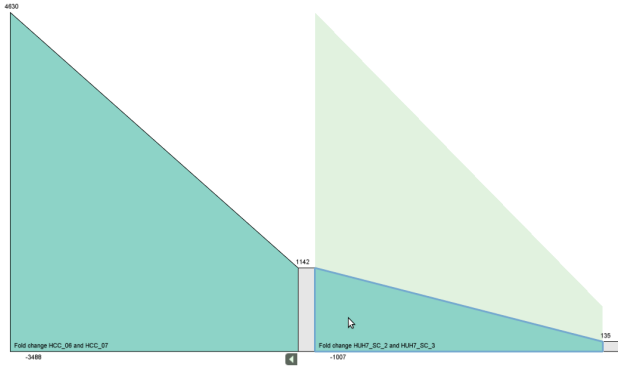

Figure 5: When hovering over the filter, its full size is shown in the background.

Having fulfilled all of the elicited requirements, we will now discuss how the described visualization technique is embedded in an information visualization framework, and give some examples on how the filter-pipeline is used.

5 Visualization Environment

The filter-pipeline view is a part of the Caleydo visualization framework[1] [9], developed at our institute. It is intended to be used for the analysis of large data sets from the life science domain, more specifically genetic and clinical data. Its multiple coordinated view system provides different ways to explore the loaded data set. For example, to explore gene expression data parallel coordinates, a hierarchical heat map, scatterplots and many further views are available. Figure 6 shows a screenshot of a typical analysis session in Caleydo.

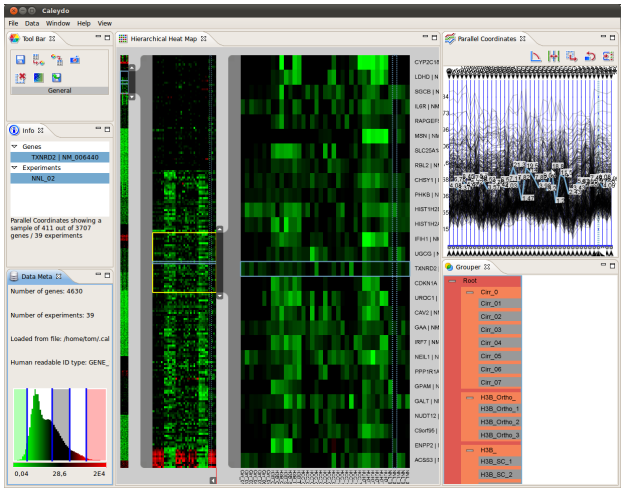

Figure 6: Example of a possible usage of Caleydo with a heat map, parallel coordinates and a view showing the hierarchical grouping of experiments.

As the initially loaded data sets in this domain are often very large, different types of filters are usually applied to reduce the size of the viewed data set, which is especially relevant to enable algorithmic methods. The different views support various ways of brushing and consequently filtering data. In the parallel coordinates, it is possible to filter data where the gene expression values never leave a given interval and therefore, for example, filter genes that are neither over- nor under-expressed. Another possibility to create a filter is to use the angular brush [6] which removes experiments with a deviation exceeding a visually specified threshold from the gene expression value of the selected gene of a specific experiment.

Caleydo also provides computational filters commonly used in gene expression analysis. One example is the fold-change filter that removes all elements which change less than a specified n-fold change to a reference experiment. Other examples are statistical variance tests, which ensure that outliers within control groups, which may be the result of errors in measurement, are removed.

The described filter-pipeline is used in Caleydo to convey the effects of complex combinations of filters. A typical scenario is shown in Figure 7.

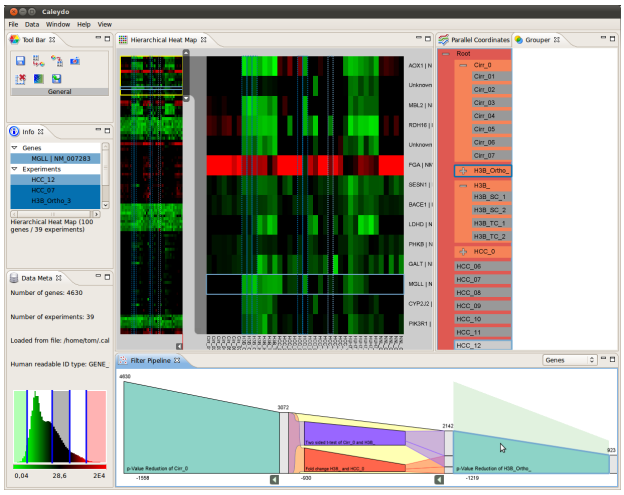

Figure 7: Example of using Caleydo with the described filter-pipeline view opened in the bottom right part of the window.

6 Use Case

In this section, we describe how Caleydo is used for the analysis of gene expression data, acquired in an experiment to find the genetic cause for liver cirrhosis. Liver cirrhosis is known to have a significant genetic component, since, for example only a portion of heavy alcohol abusers actually suffer from it, and conversely, many others, especially those with diabetes suffer from it as well. Our partners at the Medical University of Graz have found, that a specific genotype of mice (*i.e.*, a genetic variant), do not suffer from steatohepatitis, a precursory symptom to liver cirrhosis, when fed with poison over a course of eight weeks, while other genotypes do. They therefore conducted a controlled experiment with the different mouse

[1] http://www.caleydo.org

genotypes, and analyzed their genetic expression after 0 days, 7 days and 8 weeks.

For the analysis they used the Caleydo software. The first step in the analysis is to filter those genes with too much deviation within the control group, thereby ensuring statistical significance of the experiments. The second step is to filter out all genes which are not at least twice as low or twice as high in the 7 days and 8 weeks scenarios, compared to the 0 day values (by using the fold-change filter). The filter-pipeline in this case revealed that only a small portion of genes was actually dropping more than two-fold, the majority of the data was removed by the filter. A a significant part however had a two-fold increase. The biologists then investigated the remaining subset in greater detail by interactively adding, removing and refining filters.

7 Implementation Details

The Caleydo visualization framework [9] is written in Java and is based on the Eclipse Rich Client Platform (RCP)[2]. The framework is designed in a modular manner where a minimal core contains integral parts, such as the data management, the event system, *etc.*. Everything else is outsourced into separate and completely independent plug-ins which communicate with each other by using the core's message-based event mechanism.

Each view can either use the Standard Widget Toolkit (SWT)[3] to create a user-interface by using the default widgets provided by the operating system, or for graphically more advanced or three dimensional user-interfaces use the OpenGL API provided by the Java Bindings for OpenGL (JOGL)[4].

In order to synchronize all views, the data set containing the data to by analyzed is stored centrally, so that each view can access it. View changes are handled first by the view under interaction itself and then propagated to all other views which in turn update their visualization based on the new context.

The statistical filters use the R statistics toolkit [11] for calculating the filter elements which are added to the corresponding list.

8 Conclusions and Future Work

As the amount of data to be analyzed is constantly growing, filtering it is a crucial part in the processing pipeline. Therefore, it is important that an analyst is supported in understanding complex sequences as well as compositions of filters. Visualization of those filters in the proposed filter-pipeline is an ideal tool for this task. It allows to understand even complex combinations of filters, and can be modified interactively until the desired result is achieved. Visualizing a sequence of AND combined filter is straight forward, but complex combinations, like a logical OR operation applied to several filters, require much care in visualization design.

We have presented two ways of visualizing compositions of logical filters (see Figure 8 for a complex scenario with four filters combined in an OR operation). The first one, a simple, stacked rendering of filters has shown to be very cluttered and hard to understand. Consequently we developed an alternative that shows each filter in parallel contained in a compound meta-filter, thereby providing an intuitive representation of a compound filter.

Aside from these main objectives, we have elicited several minor requirements improving the interaction with such filter visualization techniques, and proposed solutions for each of the discussed points.

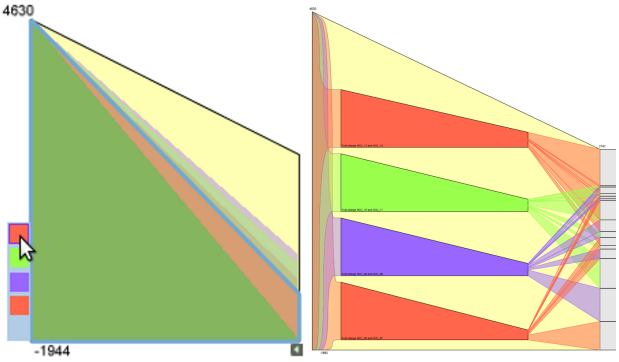

Figure 8: Comparison of different visualizations of compound filters as described in Section 4.2 using four subfilters.

The support of the complete set of logical operations as well as nested filters, for example by using zoom levels, are promising directions for future research.

9 Acknowledgments

We want to thank our partners from the Medical University of Graz, especially Dr. Karl Kashofer and Prof. Kurt Zatloukal, for providing both data and continuous feedback.

This work was funded in part by the Austrian Research Promotion Agency (FFG) through the *InGenious* project (385567).

References

[1] Christopher Ahlberg, Christopher Williamson, and Ben Shneiderman. Dynamic queries for information exploration: an implementation and evaluation. *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '92)*, pages 619 – 626, 1992.

[2] http://www.eclipse.org/home/categories/rcp.php
[3] http://www.eclipse.org/swt/
[4] http://jogamp.org/jogl/www/

[2] Richard A. Becker. Dynamic graphics for data analysis. *Statistical Science*, 2(4):355–383, November 1987.

[3] Jacques Bertin. *Semiology of graphics*. University of Wisconsin Press, 1983.

[4] Hong Chen. Compound brushing. In *IEEE Symposium on Information Visualization (InfoVis '03)*, pages 181–188. IEEE Computer Society, 2003.

[5] Helmut Doleisch, Martin Gasser, and Helwig Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *Proceedings of the Symposium on Data visualisation 2003*, pages 239–248. Eurographics Association, 2003.

[6] Helwig Hauser, Florian Ledermann, and Helmut Doleisch. Angular brushing of extended parallel coordinates. In *Proceedings on Information Visualization (InfoVis '02)*, pages 127–130. IEEE Computer Society, 2002.

[7] Daniel A. Keim, Joern Kohlhammer, Geoffrey Ellis, and Florian Mansmann, editors. *Mastering The Information Age - Solving Problems with Visual Analytics*. Eurographics, 2010.

[8] Robert Kosara, Fabian Bendix, and Helwig Hauser. Parallel sets: Interactive exploration and visual analysis of categorical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):558–568, 2006.

[9] 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 *Proceeding of the IEEE Pacific Visualization Symposium (PacificVis '10)*, pages 57–64. IEEE Computer Society, 2010.

[10] Allen R. Martin and Matthew O. Ward. High dimensional brushing for interactive exploration of multivariate data. In *Proceedings of the Conference on Visualization (Vis '95)*, page 271. IEEE Computer Society, 1995.

[11] R Development Core Team. *R: A Language and Environment for Statistical Computing*. 2010.

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

[13] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, Coneeticut, second edition, 1983.

[14] Chris Weaver. Cross-Filtered views for multidimensional visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):192–204, 2010.