

A Parallel Coordinates Style Interface for Exploratory Volume Visualization

Melanie Tory, Simeon Potts, and Torsten Möller, IEEE Member

Abstract—We present a user interface, based on parallel coordinates, that facilitates exploration of volume data. By explicitly representing the visualization parameter space, the interface provides an overview of rendering options and enables users to easily explore different parameters. Rendered images are stored in an integrated history bar that facilitates backtracking to previous visualization options. Initial usability testing showed clear agreement between users and experts of various backgrounds (usability, graphic design, volume visualization, and medical physics) that the proposed user interface is a valuable data exploration tool.

Index Terms— H.5.2 User Interfaces - Graphical User Interfaces (GUI), Screen Design, I.3.3 Picture/Image Generation - Display Algorithms, J. Computer Applications (e.g., Medical Imaging)

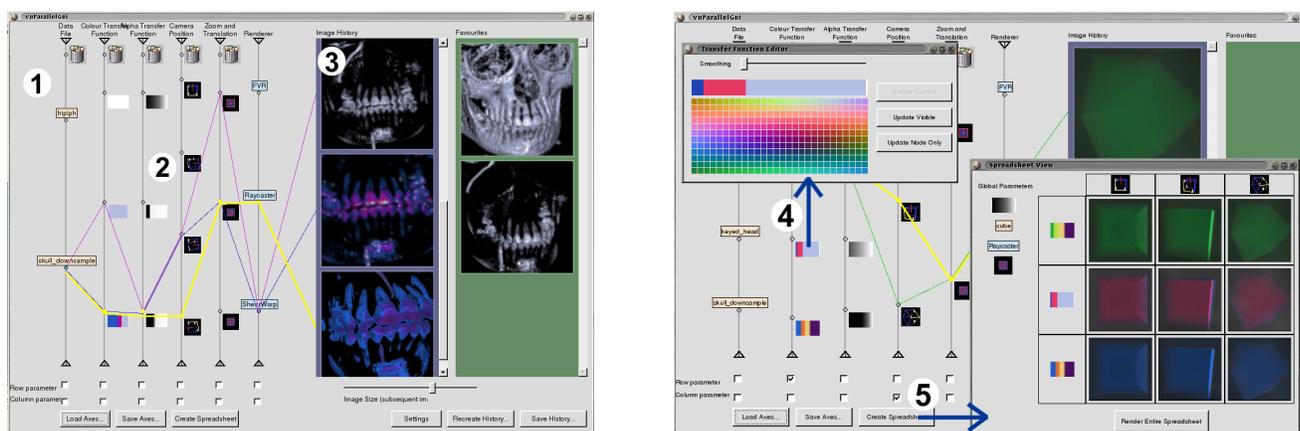


Fig. 1 Parallel Coordinates Style Volume Visualization Interface. 1. A parallel coordinates display consists of one axis for each parameter. 2. Parameter sets are represented as lines that connect the parameters and the resultant image together. 3. A history bar provides access to previous settings. 4. Existing parameter nodes are modified with an editor to create new nodes. 5. A row and column are chosen with checkboxes and a table view is created.

1 INTRODUCTION AND MOTIVATION

Data exploration and analysis are crucial steps in scientific research. Physicians, physicists, mathematicians, and other scientists examine, explore, and analyze data to gain insight into problems. The goal of visualization software is to facilitate this process in an intuitive way.

Volume visualization focuses on continuous-model data [21], often from a medical acquisition process or computational field simulation. Since the data in these fields tend to be large (several hundred MB to GB), much of the research in volume visualization has been directed towards fast and efficient rendering algorithms. In 1992, Springmeyer et al.

reported that, “While images may be the goal of visualization, insight is the goal of analysis. Visualization tools often fail to reflect this fact both in functionality and in their user interfaces, which typically focus on graphics and programming concepts rather than on concepts more meaningful to end-user scientists” [19]. Since then, interest in developing tools that facilitate data exploration, analysis, and user-tool interaction is becoming considerably more widespread [9], [17]. Although advances have been made, interfaces that support exploration and insight have substantial room for improvement, at least for volume data.

We describe a prototype interface that utilizes parallel coordinates, tables, and a history view to facilitate volume data exploration. Fig. 1 illustrates our interface. Our design uses parallel coordinates to organize and visually represent visualization parameters. To create an image, a user draws a line across all the axes to connect a specific set of param-

• Authors are with the Graphics, Usability, and Visualization (GrUVi) lab at Simon Fraser University, Computing Science Dept., 8888 University Drive, Burnaby, BC, Canada, V5A 1S6.
E-mail: {mktory, sgotts, torsten}@cs.sfu.ca

ters. Manipulating parameters changes the visualization.

The outline of this paper is as follows: §2 summarizes previous research. In §3, we describe our design in more detail. We then describe implementation details in §4 and present a usability evaluation in §5. In §6, we discuss how our interface relates to other approaches. §7 gives examples of possible applications and benefits of our tool. Finally, we suggest future research and summarize our findings.

2 PREVIOUS RESEARCH

Interfaces for volume data exploration are relatively unexplored. Traditional interfaces for direct volume rendering typically provide a simple editable graph for specifying a transfer function, as shown in Fig. 2 (label 2). A transfer function assigns a colour and opacity to each voxel based on the voxel’s data value and sometimes other inputs such as the magnitude of the gradients and second derivatives [11]. The colour and opacity determines how the volume will appear when rendered. Other rendering parameters are specified via mouse or keyboard input (e.g., for rotation and zoom level) or through graphical interface widgets.

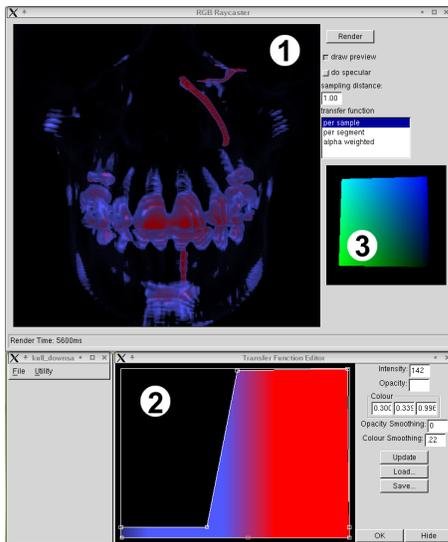


Fig. 2 Traditional Volume Visualization Interface. The interface typically consists of: 1. A rendering window. 2. A transfer function editor. 3. A graphical widget for zoom and rotation.

Searching for a good set of rendering parameters with the traditional interface is often time-consuming and frustrating because the parameter space is large. This process may be easier if the data is previously segmented. However, manual segmentation is time-consuming and automated segmentation algorithms do not exist for all applications. Furthermore, segmentation is not always meaningful (e.g. concentrations of hydrogen and oxygen gas do not have discrete boundaries and the two gases may mix). Alternate volume visualization techniques such as isosurfaces also provide a simpler search space (e.g., a few simple iso-values), but limit the range of visualization results.

One way to reduce the search space is to impose constraints on parameters as the user selects values for other parameters, to avoid useless combinations [2]. Providing a

histogram showing the distribution of data values can also help reduce the search space to meaningful settings. Bajaj et al. [1] and Kniss et al. [11] extended this idea to include additional data metrics. Kniss et al. [11] also developed specialized interface widgets for conveying this information and specifying transfer functions. In addition, they allow users to probe the data set to find data values at specific points or neighbourhoods. Similarly, König and Gröller’s interface [12] helps users relate data values in a histogram to meaningful objects in a data set via sample images. In addition, they break transfer function specification into three steps to help guide users through the process.

Another alternative is to have the computer generate many different parameter combinations and sample images, and allow the user to select useful ones or direct future searches based on images close to the desired image [6], [15]. Essentially, users explore the parameter space via visual search of resulting images.

Despite these advances in interface design and automated parameter generation, finding good parameter combinations can still be challenging, for several reasons:

- Users cannot easily keep track of what parameter combinations have been tried already.
- Undoing operations to go back to previous images requires remembering and re-entering all settings.
- Effects of distinct settings cannot be easily compared.

A few interfaces have been proposed to address the issues above. Ma [13] describes a history tool where users can easily return to previous images without remembering the parameter settings. The tool uses a graph-based display to show transfer function relationships between thumbnail views of previously rendered images. Similarly, the spreadsheet-style interface developed by Jankun-Kelly and Ma [9] allows users to explore a range of parameter combinations at the same time and compare the resulting images side-by-side. Similar spreadsheet-style interfaces have also been developed for more abstract information visualization problems [3], [20]. In a related paper, Jankun-Kelly et al. define a language for recording exploration processes [10].

Although these techniques make progress, they also have drawbacks that encouraged us to consider other alternatives. The graph could quickly grow very large, making navigation and comparisons difficult. Although spreadsheets will not grow excessively large, typically only two parameters can be explored simultaneously and navigation history is not well represented. Also, the graph-based interface does not clearly illustrate parameter combinations that have not been tried and therefore might be worthwhile to explore, and the spreadsheet-style interface does this (via empty cells) for only 2 parameters at a time. Our new parallel coordinates style interface addresses these issues and integrates naturally with the spreadsheet-style interface.

Parallel coordinates is a graphing technique that represents data dimensions by parallel axes [8], [22]. A data element is plotted by drawing a polyline that intersects each axis at the appropriate value. Parallel coordinates have been used to visualize multidimensional data for many applications. However, to our knowledge, the ideas behind parallel coordinates have not been applied to the problem of specifying visualization parameters.

3 PARALLEL COORDINATES STYLE INTERFACES

3.1 Parameter Organization and Display

Our interface provides a customizable overview of the parameter space (see Fig. 1). We use parallel vertical axes to display visualization parameters. Each point within the parameter space (i.e. a set of parameters used to produce an image) is denoted by a polyline connecting the axes.

We divide the parameters that are available in visualization software into two major categories:

- View specific parameters:
 - View position
 - Orientation
 - Zoom
 - Light position (point light)
 - Light direction (distant light)
 - Shading coefficients (ambient, diffuse, specular)
 - Rendering method (e.g., raycasting, splatting)
- Data specific parameters:
 - Colour transfer function (mapping between data value and colour)
 - Opacity transfer function (mapping between data value and transparency level)
 - Slicing plane position and orientation
 - Data set

While this division is not further explored in the implementation of this paper, we believe it is important for dealing with multivariate data. In such data sets, one will have different data-specific parameters for each variable; however, the view-specific parameters must be identical. This will be explored in our future research. In addition, this division creates a conceptual framework that allows extensions. E.g. other data manipulation tools, such as isosurface extraction or segmentation tools, need to be added to data specific parameters, not to view-specific tools.

Users create or load preset nodes for each parameter and then connect a set of nodes to produce an image. In this way, we use parallel coordinates not only as a method to display data items (parameter combinations), but also as an interface to create data items. While these items can represent simple samples of one-dimensional parameter spaces (e.g. shading coefficients), some represent samples of high-dimensional spaces. These samples can be seen as discrete entities or a notion of interpolation can be developed, helping the user to select “in-between” states. For suggestions on how interpolation for some high-dimensional parameter spaces can be achieved, see section 3.5.

Further, one can imagine wanting to split a parameter axis. For example, the colour transfer function could be split into axes for different colour components (e.g., hue, chroma, and value in the Munsell colour system). Similarly, a user could simplify the display by grouping several parameter axes into one “super-axis”. Our interface allows such split and merge operations with the base parameters of color transfer function, opacity transfer function, data set, renderer, zoom and translation, and orientation.

3.2 Integration With a Spreadsheet Interface

Spreadsheet-like layouts [9] have been shown to be useful for exploring and comparing parameter settings. Hence,

we coupled the idea to our parallel coordinates style interface, allowing users to select two axes to display as spreadsheet rows and columns, as in Fig. 1, label 5.

The history function in Jankun-Kelly and Ma’s spreadsheet was a playback animation with limited user control. We felt this limited users’ abilities to keep track of where they had been, use previous images as a basis for future exploration, and understand what parameters were used to generate previous images. We replaced this animation with an additional “parameter” in the parallel coordinates display - a time-line of all of the renderings in sequential order (see Fig. 1, label 3). Scrolling this history and selecting previous images enables users to “undo” changes, rapidly review the work they have done, and work from previous states. It also allows a non-expert to study an expert’s data-exploration process and experiment with the effects of parameter changes along the way. The history bar offers similar history functionality to an image graph [13] but has only one dimension. This has the disadvantage that branches in the exploration process are not explicitly represented. On the other hand, it provides screen space to explicitly represent visualization parameters.

3.3 Design Goals

We designed our interface to support the visualization tasks described by Shneiderman [18]. Note that we apply this visualization seeking mantra to parameter representation rather than data representation:

- **Overview:** Our design organizes all visualization parameters together in one space so that users can quickly gain an overview of the possibilities.
- **Zoom & Filter:** Visualization parameters can be easily accessed and changed to zoom and filter data sets. Selecting an image or polyline highlights the polyline in yellow so users can quickly identify the parameter settings that produced a given image. Users can simplify the parameter space (by merging axes) or fine-tune combined parameters (by expanding axes).
- **Relate:** Effects of changing parameter settings can be studied by making changes with a simple mouse action and viewing consecutive results in the history bar. Images far apart in the history can be compared by copying them to a “favourites” bar next to the history. In addition, a table layout can be created to compare all combinations of two parameters.
- **History:** Users can scroll through the history bar to see which parameter settings have been tried, and select any image to work from that previous state. On parameter axes, users can create and save nodes to store useful parameter settings (e.g. transfer functions). Returning to a previous setting simply requires dragging a line back to the node. Deleted nodes are stored in a trash container so they can be recovered. In addition, users can copy any image (with its parameter settings) to a favourites view, enabling them to continue exploration from the image later without having to re-locate it in the history bar.
- **Extract:** Each axis can be completely customized by the user to obtain a useful set of parameter settings.

Individual axes (or the entire collection) can be saved and reloaded in subsequent sessions.

Our tool explicitly represents all parameters that influence rendering. It shows current and previous settings of all visualization parameters simultaneously. Parameters can therefore be explored immediately within the context of previous settings. To the best of our knowledge this has not been tried before, and we are convinced that it offers substantial benefits to the data exploration process.

3.4 Relationship to Parallel Coordinates

Although our interface is based on the concept of parallel coordinates, it has substantial differences from ordinary parallel coordinates displays. Many of the axes do not display one-dimensional variables. For example, a transfer function is a complex concept consisting of variables such as data intensities, opacities, and sometimes intensity derivatives. Furthermore, axes can be merged to produce even more complex nodes. Sorting and interpolating nodes therefore requires slightly more complex techniques than ordinary parallel coordinates (see section 3.5). In addition, a major strength of parallel coordinates is their ability to make high-level trends in data sets apparent (by drawing many lines). In our current implementation, we did not focus on such high-level trends; instead, we display only a few lines so that users can easily distinguish line colours and relate one or a few images to their parameters. To avoid interface clutter, only the polylines corresponding to images visible in the history and an image from the favourites (if one is selected) are displayed. However, if specific needs arise, such as studying the pattern of data space exploration, it might be useful to show all explored parameter combinations at once. Our proposed interface is novel in that it is the first interface that would allow such analysis.

3.5 Sorting, Spacing, and Interpolation

We deviate from parallel coordinates in that parameters are evenly spaced on an axis by default, and new nodes are added to the bottom of the axis. This chronological ordering allows users to easily remember the positions of previous nodes and locate new nodes as they are created. Users can customize the display by repositioning axes (shifting them one position left or right). For example, if a user wanted to select the renderer before other parameters, the renderer axis could be moved to the left. Users can also drag nodes to rearrange them on an axis, or automatically make the spacing even. Nodes on an axis can be untangled based on the weighted position of adjacent connected nodes, or sorted based on a numerical measure (l_2 difference) of how much they differ from a chosen node.

We define, but did not implement, interpolation between two nodes. While the meaning of such interpolation should be established depending on the specific application, we believe that for most volume rendering parameters, linear interpolation of individual values would be effective (e.g., for opacity transfer functions, linearly interpolate the opacity value for each individual data value to produce a new graph). We suggest quaternions for interpolating between orientations. Interpolation between renderers could possibly be a simple image blending their outputs.

Utility of such interpolation may be limited, but may be useful for blending variables in multivariate data sets. One can also imagine interpolating (morphing) between two data sets. Such interpolation could be valuable for time-series volumes or other related data sets. For time series data, linear interpolation would be effective if the data sets are the same size and are registered. Resampling and/or more complex methods may be needed to interpolate between data sets with less similarity. In addition, the meaning of an interpolated data set (in the context of the application) should be clear before it is used for data analysis.

4 IMPLEMENTATION

We implemented a parallel coordinates style interface for volume rendering. The interface is built with QT [16], a windowing toolkit, and selected rendering tools from vu-Volume, a volume visualization suite developed in our lab.

4.1 Interface Components

Fig. 3 illustrates our interface. A set of parallel axes forms the central component (vertical lines in Fig. 3). Discrete nodes can be added to each axis from a popup menu command or by manipulating a previous node to produce new parameter values. Parameter values can be saved for a future session so that the same creation process does not have to be repeated every time the software is run.

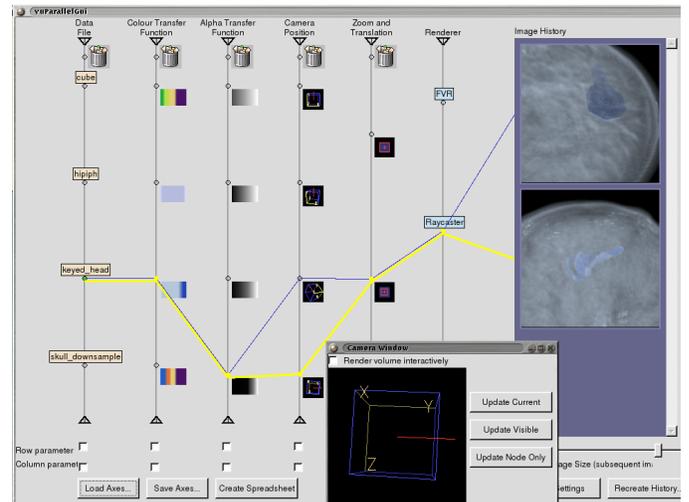


Fig. 3 Setup of the parallel coordinates style interface. The parallel coordinates organization is shown in the background, with an orientation editor in the foreground.

Immediately to the right of the axes is the history view, containing images that are created. The set of axes and the history view are all that is needed to create, edit, and visualize any number of parameter combinations. A combination of parameters is created by selecting one parameter value from each axis, by either clicking on individual nodes until the set is complete or by dragging the mouse across all axes from left to right. Dragging selects the node closest to the pointer as the axis is crossed, reducing the need for careful aiming. A series of line segments connect parameters together as they are selected; when the set of parameters is complete the chosen renderer (which itself is a pa-

parameter) is invoked with the remaining parameters as arguments. The renderer returns an image that is displayed in the history view with a line connecting it to its parameters.

At this point, additional parameter sets and their rendered images can be created in a similar fashion. Alternatively, exploration can be continued from any previous parameter set by either editing one of its parameters and clicking an “Update” button (as was done with the camera position parameter in Fig. 3), or by clicking and dragging the line that links the set of parameters together to a new node on one of the axes. Following either of these actions, a new image is added to the history column and connected to its rendering parameters via a new polyline. Each polyline is assigned a unique colour so that users can see at a glance which parameters were used to create each image. Additionally, it is easy to determine how two or more renderings differ in terms of parameters. For example, the two renderings in Fig. 3 are identical except for camera position.

We implemented the following parameters: data file, colour and alpha transfer functions, camera orientation, zoom, translation, and volume rendering method (renderer). Lighting, shading, and slicing planes were not included because manipulation of these parameters is currently not standardized in *vuVolume*. We intend to add these parameters as they are implemented.

4.2 Additional Features

We implemented various additional features to make the process of exploring data more flexible and powerful.

By default, every change produces a new image. To prevent the history from becoming very large, we included options to store an image in the history only every n (where $30 \leq n \leq 150$) seconds when changing parameters, or only if it differs enough from the last image (measured by summed difference squared over the image pixels).

Trashcans at the top of the axes (tops of Fig. 3) provide a way to dispose of unwanted nodes without permanently destroying them. Trashed nodes can be viewed and restored by opening a trashcan.

Axes can be merged, an operation that could be useful for a pair of similar parameters such as opacity and colour components of a transfer function, as in Fig. 4, label 1. However, we did not limit merging to any such specific cases. When axes are merged, the parameters combined in a node are edited and updated in a single window. Composite axes can be expanded into components again to treat parameters separately. When axes are merged, we pair connected nodes together, and pair unused nodes with a default node. When an axis is expanded into components, we split every node and eliminate any duplicate nodes that are formed. This treatment of merging is sufficient to keep the number of nodes from growing too large on a merged axis while preserving all of the parameter values created if the axes are expanded again. We represent merged nodes by composite icons so users can easily see which axes are merged (e.g., see the opacity and colour transfer function icons near label 1 in Fig. 4). When an axis is split, the composite icons are broken into their visually distinguishable components so users can easily locate each new axis.

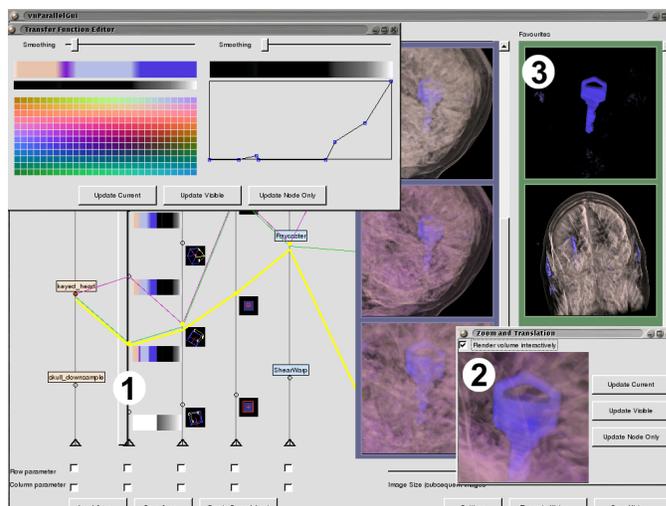


Fig. 4 Additional Features: 1. Merging axes: colour and opacity transfer function axes are merged. Nodes are now edited via a combined editor, shown at the top left. 2. Interactive zoom / translation: activating the check box displays the data set as it is zoomed. 3. Favourites view.

A useful feature supported by many visualization applications is interactive zoom, translation, and rotation. We included this feature, interactively rendering images of the current data set in the rotation, zoom and translation widgets when the user checks a checkbox (as with the zoom parameter in Fig. 4, label 2). This could be extended to interactive transfer functions, lighting, cutting planes, and any other parameter except renderers and data sets. Note that we did not include interactive transfer function editing at this time because the renderers we were working with did not all render at sufficiently interactive rates. However, we consider interactive transfer function editing important and suggest it could be achieved via progressive refinement, specialized hardware, or by using a faster rendering technique while the transfer function is modified and then switching rendering methods when the transfer function is unchanged for some time.

A Favourites view on the far right of the interface complements the exploration history by allowing users to make copies of images that they deem important, want to keep nearby for later exploration, or want to compare side-by-side (see Fig. 4, label 3). Thus, while the history bar keeps track of everything the user has done (in chronological order), the favourites bar provides a customizable space.

We also included controls to load and save axes individually or as a set, and to save and reload the entire history or just a single set of parameters. Reloading a history re-renders each image in sequence, adding nodes to the axes as needed (without adding duplicates). A larger rendering window can be opened by double-clicking on any image, producing the same rendering at a higher resolution and providing the user with a button to save the image.

4.3 Tables

We expected the spreadsheet-style interface [6] to be useful for side-by-side comparisons, and included the basic functionality as a branch of our interface called tables (see Fig.

5). A table is created by selecting a row parameter and a column parameter via checkboxes located below the parameter axes (see Fig. 5, label 1).

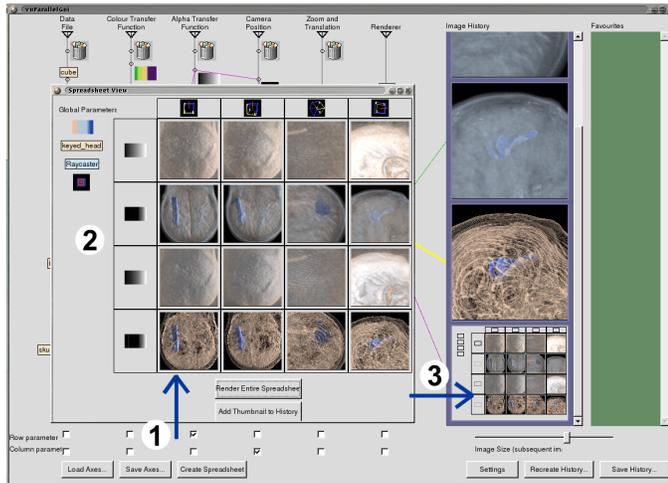


Fig. 5 Example of a table. 1. Two axes are selected via check boxes. 2. A table is created from the selected parameters. 3. A thumbnail view of the table is copied to the history bar.

In contrast to Jankun-Kelly and Ma’s spreadsheet [9], we currently do not consider cell operations (e.g., union or intersection of parameters) or scripting. We implemented the spreadsheet to illustrate how it could integrate with a parallel coordinates style interface. Due to the development work involved, we currently do not support these spreadsheet-like actions and refer to these views more accurately as “tables”. However, we do believe that these sophisticated features would enhance spreadsheet usability.

Each table is created in its own window, using the current parameter set for global parameters. Parameters can then be edited, rows and columns can be added, and users can choose a global parameter to use as a new row or column parameter. Individual images can be rendered by clicking in the corresponding table cell, or all cells can be rendered at once with a single button.

We connect tables to the parallel coordinates view by allowing users to (1) copy a selected image from the table into the parallel coordinates view to work with it, and/or (2) place a thumbnail of the rendered table in the history column (see Fig. 5, part 3). The thumbnail is connected to its global parameters by a polyline and can be opened later, keeping the same global, row, and column parameters.

5 USABILITY EVALUATION

To evaluate our interface and compare it to table-based and traditional interfaces, we conducted an initial usability evaluation based on inspection techniques [14]. We collected quantitative rating scale data on several heuristics that were designed to determine whether the parallel coordinates style interface met our objectives (helping users understand and relate parameter settings in the context of data exploration). The heuristics also provided some structure to discussions with participants. However, our main objective was to explore qualitative issues such as

strengths, weaknesses, and usefulness of each interface and ideas for future development. These results would determine what issues we should consider before conducting a more complex future study. Thus, we consider our qualitative results more valuable than the rating scale numbers. Future studies could expand on our results, providing a more comprehensive assessment of these interfaces (and more specialized ones) in the context of real applications.

5.1 Evaluation Procedure

Five experts assessed the interfaces using heuristic evaluation techniques [14]. In most heuristic evaluation procedures, interface design experts review an interface with respect to standard interface guidelines. We felt that this process would limit our results to interface details and would not expose deeper issues such as how well the interfaces support data exploration. Thus, we developed our own heuristic guidelines based on the questionnaire for user interface satisfaction [4] and common visualization tasks [18]. In addition, we chose experts from a wide variety of backgrounds, including usability, volume visualization, and graphic design. We also included one end user (a physicist with expertise in medical imaging). We expected this distribution of experts to provide a wide variety of opinions and insight into the tools. None of the experts had prior involvement with the parallel coordinates project.

Experts conducted evaluations separately. They were first introduced to the interfaces and the concept of volume rendering. Data sets and default parameter values were set up ahead of time. Experts were asked to perform two sample tasks: (1) explore the data sets and (2) search for an identifiable object (a key) in one data set. Searching for the key was included to provide a directed exploration task since our participants were not end users with their own goals. During the evaluation, the experimenter used contextual inquiry techniques to encourage participants to discuss their actions, ideas, and opinions (including issues such as how and when the interfaces might be useful). Participants also completed a written report containing questions about advantages and disadvantages of the interfaces and 7-point rating scales for the following heuristics:

- 1) **Learning:** Ease of learning to use the interface
- 2) **Exploring:** Ease of exploring data
- 3) **Parameter Space:** Ease of understanding the parameter space (rendering options)
- 4) **Finding Combinations:** Ease of finding good parameter combinations (images)
- 5) **Changing Parameters:** Ease of changing parameters
- 6) **Relationships:** Ease of comparing effects of parameter settings
- 7) **Keeping Track:** Ease of keeping track of what you have done
- 8) **Undo:** Ease of going back to previous settings
- 9) **Organization:** Clarity of information organization
- 10) **Tasks Straightforward:** How often tasks can be performed in a straightforward manner
- 11) **User Needs:** How often both experienced and inexperienced users’ needs are considered

5.2 Evaluation Results

Fig. 6 shows average ratings from the heuristic evaluation. Qualitatively, one can see that the table and parallel coordinates style interfaces rated higher than the traditional interface, with one exception: the traditional interface was perceived as simpler and therefore easier to learn.

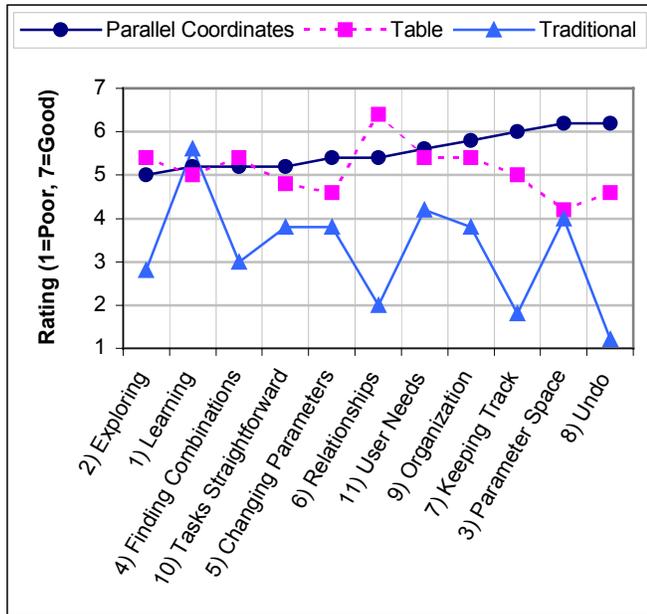


Fig. 6 Average ratings in the heuristic evaluation. Heuristics are sorted by increasing average rating for the Parallel coordinates style interface.

Parallel coordinates and table interfaces had different strengths. Tables were considered particularly good for studying relationships (comparing effects of parameter settings) because images based on a large number of setting combinations could be viewed side-by-side. For the same reason, this layout was also quite useful for exploring data sets. However, the parallel coordinates style interface was rated better for understanding the parameter space, changing parameter values, and history functionality (undo and keeping track of where you have been). Because these two interfaces were useful for different purposes, all five evaluators preferred the parallel coordinates / table combination to either interface alone or the traditional interface.

Rating data was analyzed by multivariate repeated-measures analysis of variance (ANOVA) with the Huynh-Feldt correction. For details on this technique, see [7] or another statistical text. The overall multivariate difference between interfaces was not significant (Wilks' Lambda $F(16,2) = 10.3, p = 0.092$). However, there were significant main effects for exploring ($F(1.3, 5.2) = 15.5, p = 0.009$), finding combinations ($F(2,8) = 11.3, p = 0.005$), changing parameters ($F(2,8) = 4.6, p = 0.047$), relationships ($F(1.8, 7.1) = 34.7, p < 0.001$), keeping track ($F(1.2, 4.9) = 49.8, p = 0.001$), undo ($F(2,8) = 29.6, p < 0.001$), organization ($F(2,8) = 21.0, p = 0.001$), and user needs ($F(1.5, 6.1) = 6.4, p = 0.037$). With the exception of user needs (where no pairwise significant differences were found), pairwise comparisons showed that the traditional interface was significantly different from either or both of the other interfaces, but the parallel coor-

ordinates and table interfaces were not significantly different from each other. Based on our raw data, we would expect to see significant differences between these two interfaces if more evaluators were included in the process.

Evaluators felt the interfaces would be useful for different purposes. Table 1 outlines the uses they identified.

TABLE 1 MAIN USES OF EACH INTERFACE

<i>Parallel Coordinates</i>	<i>Table</i>	<i>Traditional</i>
Unknown data sets	Comparison of many settings	Well-known data sets
Initial (coarse) exploration of many options	Fine-tuning parameter settings	Simple tasks
History functions	If only two parameters are important	Novice users (easier to learn) ¹

¹ However, one evaluator felt the traditional interface would be terrible for novices because there was no undo function.

All evaluators agreed that the parallel coordinates / table combination was the overall best interface. They felt the traditional interface would only be useful for very specific situations (e.g., for an expert user who wants to quickly view a well-known data set or, possibly, to help a novice user learn the basics of volume rendering). Desirable features in the parallel coordinates style interface were:

- Ability to “see at a glance” what options were available. Evaluators felt the traditional interface had fewer options and some options were hidden.
- Ability to go back to previous states.
- Dragging the mouse across the axes to create a new image. This method was not intuitive at first, but evaluators quickly learned the technique and appreciated its speed and gesture-like quality. The alternate multiple-click method was slow and cumbersome.
- Looking at the polyline to identify the set of parameters for a selected image (to see how it was created).
- Copying images to favourites for personalized organization or to compare images far apart in history.

Experts disagreed somewhat on how the parallel coordinates and table interfaces would be used together. Two experts suggested the parallel coordinates style interface would be used initially for coarse exploration (to find a set of parameters that produces an image close to the desired goal); the table would then be used to fine-tune the parameters since a wide array of variations could be quickly tried. By contrast, two other evaluators felt the table would be useful for initial exploration (to quickly scan a large combination of images based on default settings); interesting images could then be moved to the parallel coordinates style interface for further examination and alteration.

One evaluator felt the main advantage of the parallel coordinates / table interface over the traditional interface was the history function. He believed that adding a one-level undo function to the traditional interface would produce 50% of the added benefit of the other interfaces. Although he chose parallel coordinates / table as the best overall, he

liked the traditional interface because functions were integrated. For example, zooming and rotating were done on the rendered image (via mouse buttons). Evaluators felt that preset camera settings would be important for comparing well-known data sets from standard orientations, but interactive camera manipulation was more useful and intuitive for exploration. Also, colour and opacity transfer functions were integrated into one window. This integration saves screen space, allowing more space to be devoted to drawing images. It also reduces the cognitive overhead and time required for users to switch from one tool to another. One evaluator felt the parallel coordinates style interface should contain a “streamlined” mode similar to the traditional interface, where users could set default values for parameters they are not currently interested in and hide axes for those parameters to save screen space and reduce clutter. Such a streamlined mode could be created by allowing rotation / zoom / translation directly in the images in the history bar, by creating more integrated editors when axes are merged, and/or by allowing users to work with a traditional interface while the parallel coordinates style interface runs in the background and saves a history. We plan to explore these ideas in future work.

Evaluators made several other suggestions that would be interesting to explore. These included:

- An option to switch between chronological ordering and custom ordering of items in the history bar.
- “Garbage collection” routines so users do not need to manually remove unwanted notes.
- A visual distinction between preset nodes (which may have been carefully set up by an expert for a specific data set) and nodes created by the user.
- Interactive interpolation so dragging the mouse between nodes progressively changes the image.
- Closer integration between parallel coordinates and table interfaces (e.g., ability to select a node in the parallel coordinates style interface to add to the table).

Overall, reactions to our parallel coordinates style interface were very positive, indicating that it is a promising idea for volume exploration. Our evaluation process identified several directions for future development and research. We should note that our usability study has several limitations. Because it was an initial evaluation with experts, we believe further testing should be done with end users in specific application areas. Furthermore, the interface was tested with a simplified and integrated version of the spreadsheet style interface. Future studies could compare a parallel coordinates style interface with a more complete spreadsheet implementation or with other visualization interfaces such as design galleries [15].

6 DISCUSSION

Interfaces compel users to interact with data and ideas in various ways. For example, design galleries [15] encourages an image-based interaction style, whereas König and Gröller’s interface [12] encourages a three step transfer function editing style. Our interface emphasizes parameters and their relationships, similar to spreadsheet style [9] and graph-based [13] interfaces. Each manner of interaction is

suited to different tasks. We expect image-based approaches to be useful for initial exploration and to produce appealing images. However, to relate rendered images to data values, users will often need to visualize parameters, especially transfer functions. A transfer function editing style may be best when the user is very familiar with the data set and has a concrete task (e.g., a radiologist searching for a tumour). By contrast, interfaces that emphasize parameters and their relationships will be useful for exploring data (to understand how different avenues of exploration relate) and to compare different data sets or renderers.

Screen space usage should match the goals of an interaction style. Image-based approaches devote most of the screen real estate to rendered thumbnail images so users can quickly scan a large selection of images. Transfer function editing approaches devote approximately half the screen to visualizing / editing the transfer function, and half to a large rendered image. Because we focus on parameters and relationships, our interface devotes a large portion of the screen to parameter visualization. This could be considered a disadvantage in situations where users want to focus on the data rather than parameters. However, in these circumstances, our interface could integrate with other methods. For example, the parallel coordinates style interface could run in the background, saving a history of images and parameters, while users work interactively on a large image. The parallel coordinates style interface would be brought to the foreground when users want to study parameters, return to previous states, or compare data sets, renderers, or other settings.

Other data manipulation techniques that enhance the exploration process can be added to the interface. For example, a 2D slicer could be added as a data-specific parameter axis, an axis that modifies the data before rendering. One could also imagine incorporating more sophisticated segmentation methods as additional axes. This might be useful to researchers who want to compare effects of different segmentation algorithms, or to doctors who use different segmentation tools for different body parts. Essentially any data manipulation tools (e.g. slicers, segmentation tools, iso-surfaces, morphing tools, etc.) could be integrated; however, the goal of our paper was to show a proof of concept that the explicit layout of the parameter space can enhance the data exploration process. For specific applications many different parameters can be added to this framework or removed – depending on whether they are found to be useful for the exploration task at hand.

7 POTENTIAL APPLICATIONS AND BENEFITS

We envision a great impact of this tool in many areas of scientific research that rely on the visualization of large or complex data sets. Parallel coordinates can be used to explore the parameter space for almost any type of visualization task once it is applied to the particular domain.

At first we plan to integrate it into tools that are being used by our collaborators in the Vancouver General Hospital. As an initial step in this direction, we included a medical imaging physicist from this group in our evaluation process to determine how our interface should be custom-

ized for this application. This evaluator felt the parallel coordinates style interface (with minor modifications and integration into a comprehensive imaging system) could be valuable for comparing nuclear medicine heart data sets and finding appropriate threshold levels for heart visualization. We also plan to work with colleagues in applied mathematics, using the tool to study fuel cell simulations.

In addition, we are finding the interface useful in our own lab, for comparing output of different volume rendering algorithms and implementations. With our traditional interface (see Fig. 2), algorithms could not be easily compared without saving and reloading transfer functions and manually setting view parameters such as zoom and rotation. With the parallel coordinates and table interfaces, a set of parameters can be rapidly duplicated and applied to a different renderer; the two output images can then be compared side-by-side. Thus it is easier to identify bugs and examine strengths and weaknesses of rendering algorithms.

8 FURTHER RESEARCH

Several avenues for further research are apparent. Linear axes do not adequately represent the parameter space for multi-dimensional parameters such as transfer functions. Our solution was to treat nodes as discrete, unrelated entities, but we believe other mappings from high-dimensional spaces to linear axes might be possible. In addition, multivariate data (data with more than one dependent variable, such as pressure and temperature in some environment) is typically a challenge for visual data exploration. We believe a parallel coordinates style interface is well suited for this task because it can handle the large number of parameters associated with data from complex domains. Data specific parameters for each modality could be replicated and added to the interface as additional axes.

Our interface could also be integrated with data exploration tools described by others. For example, parallel coordinates could be used to select parameters for exploration at a higher level through Design galleries [15]. Node operators described by Jankun-Kelly and Ma [9] (union, intersection, etc.) could be incorporated to simplify node creation. Finally, the framework described by Jankun-Kelly et al. [10] could be used to store the visualization history and relationships between history items, allowing more flexibility in terms of how the history is replayed or visualized.

Issues of scalability inevitably surface, arising from a limit to screen real estate but no apparent upper bound on the size of a history that could be created, the number of visualization parameters, the number of axes, or the number of nodes. These issues would become more pronounced when we consider multivariate data, because the number of visualization parameters increases. Allowing users to merge axes and set default values / hide axes helps to reduce screen space clutter. Focus + context techniques such as fisheye views [5] could also alleviate the problem somewhat. However, these approaches do not completely solve the problem and the scalability issue remains open.

9 CONCLUSION

We presented a parallel coordinates style user interface that facilitates volume data exploration. All parameters are clearly organized and visible so that users can see and remember what options are available and what settings generated a given image. A history bar allows users to easily backtrack to previous states and quickly scroll to see which options have been previously tried. Initial usability testing showed that the proposed user interface is a promising tool for exploration of continuous model data.

10 ACKNOWLEDGMENT

Thanks to T.J. Jankun-Kelly and K.L. Ma for sharing their VisSheet code and providing valuable feedback. We also thank our evaluators and members of the GrUVi lab for useful discussions. Images were rendered using vuVolume, a volume rendering framework developed in the GrUVi lab. The data set in Fig. 3-5 is courtesy of Jeff Orchard, and data sets in Fig. 1-2 are from volvis.org. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

11 REFERENCES

- [1] C.L. Bajaj, V. Pascucci, and D.R. Schikore, "The Contour Spectrum," *Proc. IEEE Visualization*, pp.167-173, 1997.
- [2] L.D. Bergman, B.E. Rogowitz, and L.A. Treinish, "A Rule-Based Tool for Assisting Colormap Selection," *Proc. IEEE Visualization*, pp. 118-125, 1995.
- [3] E.H. Chi, J. Riedl, P. Barry, J.A. Konstan, "Principles for information visualization spreadsheets," *IEEE Computer Graphics and Applications*, vol. 18, no. 4, pp. 30-38, 1998.
- [4] J.P. Chin, V.A. Diehl, and K.L. Norman, "Development of an Instrument Measuring User Satisfaction of the Human-Computer Interface," *Proc. CHI*, pp. 213-218, 1988.
- [5] G.W. Furnas, "Generalized fisheye views," *Proc. CHI*, pp. 16-23, 1986.
- [6] T. He, L. Hong, A. Kaufman, and H. Pfister, "Generation of Transfer Functions with Stochastic Search Techniques," *Proc. IEEE Visualization*, pp. 227-234 and 489, 1996.
- [7] S.W. Huck, *Reading Statistics and Research*, 3rd ed. Addison Wesley Longman: New York, 2000.
- [8] A. Inselberg and B. Dimsdale, "Parallel coordinates: A tool for visualizing multidimensional geometry," *Proc. IEEE Visualization*, pp. 361-378, 1990.
- [9] T.J. Jankun-Kelly and Kwan-Liu Ma, "Visualization Exploration and Encapsulation via a Spreadsheet-like Interface", *IEEE Trans. Visualization and Computer Graphics*, vol. 7, no. 3, pp. 275-287, 2001.
- [10] T.J. Jankun-Kelly, Kwan-Liu Ma, and Michael Gertz, "A Model for the Visualization Exploration Process," *Proc. IEEE Visualization*, pp. 323-330, 2002.
- [11] J. Kniss, G. Kindlmann, and C. Hansen, "Interactive Volume Rendering Using Multi-Dimensional Transfer Functions and Direct Manipulation Widgets," *Proc. IEEE Visualization*, pp. 255-262 and 562, 2001.
- [12] A. König and E. Gröller, "Mastering Transfer Function Specification by Using VolumePro Technology," *Proc. 17th Spring Conf. Computer Graphics*, pp. 279-286, 2001.
- [13] K.-L. Ma, "Image Graphs: A Novel Interface for Visualization Exploration," *Proc. IEEE Visualization*, pp. 81-88, 1999.
- [14] R.L. Mack and J. Nielsen, "Usability Inspection Methods: Executive Summary", Reprinted in R.M. Baecker, J. Grudin, W.A.S. Buxton, and S. Greenberg, eds., *Readings in Human Computer Interaction: Toward the Year 2000*, Morgan Kaufmann Publishers, San Francisco, pp. 170-181, 1995.
- [15] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber, "Design Galleries: A General Approach to Setting Param-

- ters for Computer Graphics and Animation," *Proc. SIGGRAPH 97*, pp. 389-400, 1997.
- [16] Qt windowing toolkit:
<http://www.trolltech.com/products/qt/index.html>.
- [17] P. Rheingans, "Are We There Yet? Exploring with Dynamic Visualization," *Visualization Viewpoints article in IEEE Computer Graphics and Applications*, vol. 22, no. 1, pp. 6-10, 2002.
- [18] B. Shneiderman, "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations," *Proc. IEEE Symp. on Visual Languages*, pp. 336-343, 1996.
- [19] R.R. Springmeyer, M.M. Blattner, and N.L. Max, "A Characterization of the Scientific Data Analysis Process," *Proc. IEEE Visualization*, pp. 235-242, 1992.
- [20] C. Stolte, D. Tang, P. Hanrahan, "Polaris: a system for query, analysis, and visualization of multidimensional relational databases," *IEEE Trans. Visualization and Computer Graphics*, vol. 8, no. 1, pp. 52-65, 2002.
- [21] M. Tory and T. Möller, "A Model-based Visualization Taxonomy," in preparation. Initial draft available as a technical report: *SFU-CMPT-TR2002-06*, Computing Science Dept., Simon Fraser University, 2002.
- [22] E. Wegman, "Hyperdimensional Data Analysis Using Parallel Coordinates," *J. of the American Statistical Association*, vol. 85, no. 411, pp. 664-675, 1990.

Melanie Tory is a PhD candidate in the Graphics, Usability, and Visualization lab at Simon Fraser University. Her research objective is to enhance the value of visualization tools by developing and evaluating effective display designs, interaction techniques, and user interfaces. Melanie received a BSc degree from the University of British Columbia in 1999, and expects to complete her PhD degree in 2004.

Simeon Potts is a BSc candidate in Computing Science at Simon Fraser University in his senior year. His involvement in this project came from an interest in graphics and visualization research following course work in computer graphics.

Torsten Möller is an assistant professor at the School Of Computing Science at Simon Fraser University. His research interests include the fields of Scientific Visualization and Computer Graphics, especially the mathematical foundations of visualization and graphics. He is co-director of the Graphics, Usability and Visualization Lab and serves on the Board of Advisors for the Centre for Scientific Computing at Simon Fraser University. He has been appointed Vice Chair for Publications of the IEEE Technical Committee of Visualization and Graphics. He received his PhD in Computer and Information Science from Ohio State University in 1999 and a Vordiplom (BSc) in mathematical computer science from Humboldt University of Berlin, Germany.