Interactive Hierarchical Data Investigation using Abstractions – the Waltz Visualization Environment

Dr. Jonathan C. Roberts

Computing Laboratory,
University of Kent,
Canterbury,
Kent, CT2 7NF, UK
J.C.Roberts@ukc.ac.uk

Technical Report 24-96

December 19, 1996

Abstract

Scientific visualization is becoming common in the work-place; hastened by the growth in the power of computers generating data that needs to be visualized. There is a desire to generate quick representations of the data, that can be used to select and specialize the data into smaller manageable sections.

This paper describes a visualization environment (Waltz) where the user can segment the data into interesting features, select a subset of the features, interact and manipulate the images between views: controlled by the notion of linked abstractions. An abstraction is a view of the data in which the data elements are transformed or augmented to facilitate the understanding of the data.

1 Introduction and Motivation

Many popular visualization systems [25, 1, 34, 3, 29] are based on the data-flow paradigm. These environments usually provide a visual programming interface, where the user connects a suite of modules that filter, map and display the data. These systems are extensible and adaptable: as user defined modules can be incorporated and used with existing modules. However the visualization process is often long and involved, requiring many modules to achieve a correct visualization.

Other visualization environments provide a suite of classes, types and functions that are programmed, interpreted or compiled to produce the visualization result [9, 16, 23, 14, 5]. These environments are powerful visualization tools, but require programming and specific expertise to generate the visualization image.

Our aim is to generate a visualization system that will shorten the visualization cycle, in that the user can immediately depict a view of the data, understanding how to generate additional views and dissections of the data. We believe that this can be achieved using the concept of abstractions in the design and implementation of a visualization system.

In this paper we define an 'abstract image': abbreviated as an abstraction (section 2); explain how the abstract notion is incorporated with the generalization and specialization processes to form the Waltz visualization Metaphor (section 3) describing the modules and user-interface to aid the data investigation using the paradigm. Within section 4 we review related work, section 5 describes some examples using the Waltz environment and the paper finishes with a discussion and conclusion about the described work.

2 Abstractions

A book illustrator wishing to describe and depict a figure of a microwave oven, for example, can present the information in many guises: as an actual photograph, a photograph of the component parts, a simplified schematic-drawing or a circuit diagram of the main electronic circuitry. Each of these *diagrams* depict a view of the oven and can be used to illustrate a specific characteristic. To fully understand the whole object, we claim, that the object should be explained and depicted using a number of these different methods.

Likewise, data visualizations can be generated by many methods and displayed in multiple forms. For example, a surface can be generated over a three dimensional object in either a high or low level of detail. The detailed display provides precise information about the data, often taking time to understand and interrogate the data. However, the simple form provides an overview of the data but at the loss of detailed and precise information. We believe that there is a need to display both styles of images: at both levels of abstraction.

We define an abstraction to be: views that are related to the original view but have been altered or generalized to simplify the image. An abstraction can lose data (in a controlled manner) in order to express the information in a clearer and simpler way [18].

Any visualization (on a computer screen) could be seen as an abstraction of the original data because the data set is approximated as an image by projections, scalings and transformations. Most visualizations are created with a direct spatial mapping of each position in the data onto the screen using, say, a perspective projection, but this is "less of an abstraction" than a projection that does not preserve the exact positional information of the spatial data. Haber and McNabb [7] state "suitable nonlinear mappings can be more effective in revealing subtleties of structure".

The Waltz system provides methods to segment (generalize) the data into spatially-connected elements (named groups), select a subset (specialization) of the groups and display the results in multiple views (abstractions). The abstractions (for a given specialization) are inherently linked together and provide methods to directly manipulate and select the groups for specialization; where an abstraction can be used to control the orientation and appearance of another abstract view.

3 The Waltz Metaphor

The abstract views provide multiple ways to view the data; however, some of these views are difficult to interpret in isolation. Therefore, there is a requirement to correlate the information within each view to other views. Waltz achieves this through both object and image based correlation. Each process is described below and expanded within the following sections.

The object correlation is provided by dividing individual data elements into similar groups and linking the group information between views. We name this grouping process *generalization* (section 3.1), as the data can be classed as becoming more general. Operations over the groups of elements can include merging groups or selecting a subset of groups. The latter operation describes the *specialization* operation, where the data becomes more specialized (see section 3.2). Waltz incorporates the generalization and specialization operations to aid the abstraction process. The Waltz user generalizes the data into groups, displays the information as abstractions (section 3.3) and selects a subset of groups (making the specialization). Additional generalizations, abstractions and specializations can be performed on the specialized data.

The Generalization, abstraction and specialization processes form a complete visualization system and can be compared to the filter, map display of the traditional dataflow paradigm.

Waltz stores the original data in the Data File Module (section 3.4) and the three-processes are contained within the Grouped-Abstraction-Module (section 3.5), Figure 1. Each Grouped-Abstraction-Module contains (1) input and output ports that can be connected to any other Grouped-Abstraction-Module, (2) a generalization method that automatically inherits generalizations from previous levels or can generate new groups, (3) abstraction methods that automatically display the current data and (4) a specialization process that can generate a new data-specialization at the request of a user.

Therefore, the specialization process provides a hierarchical segmentation method where the data can be segmented and consequently further subdivided. This hierarchy is a tree; Waltz explicitly displays this hierarchy within the layout and position of the modules on the canvas (section 3.6).

Image-based correlation is provided through Linked-manipulation (section 3.7), linking the control manipulation between abstract views. Operations are provided to highlight specific groups and directly manipulate the

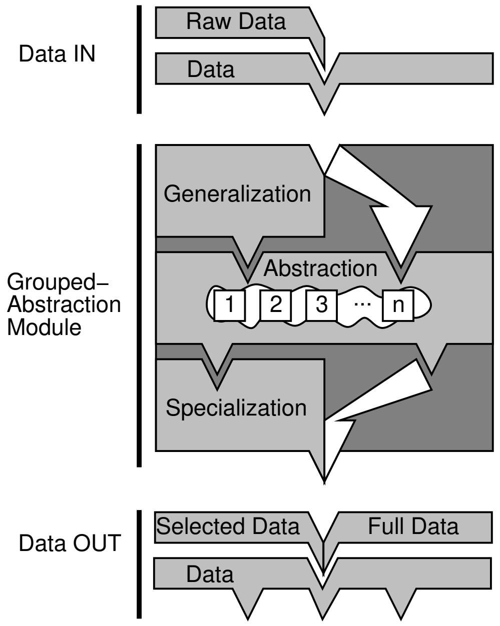

Figure 1: Generalization, Abstraction and Specialization

abstract-views.

We describe each aspect of the visualization metaphor, within Waltz, in more detail in the following sections.

3.1 Waltz Generalization

Waltz segments the data into 'similar' regions called generalizations. Bertin [2] describes a generalization as the spatial equivalent of simplification and that "Generalization thus involves discovering concepts which are applicable to the available signs and which are defined as similar for a certain area, such that this area can be considered as different from neighboring areas". These similar regions within a generalization are named *groups*.

Generalization in Waltz is achieved using a recursive flood filling algorithm. The algorithm proceeds by flooding to every adjacent voxel where the difference in value is less than a fixed threshold; this algorithm spatially segments the voxels into groups of similar values. A merging stage then joins the smaller sized groups into appropriate adjacent groups (similar to [12]).

3.2 Waltz Specialization

The Specialization process makes the data smaller and more specialized. Waltz provides this mechanism through user selection: where the user selects (Highlights) a subset of the generalized groups.

The specialization process generates a hierarchical data simplification model using the (algorithmic) generalization and (user controlled) specialization functions; the hierarchy being implicitly displayed by the layout of the modules (see section 3.6).

3.3 Waltz Abstractions

An abstraction is a view that displays only certain aspects of the information from the data set. The view, therefore, hides or loses information to facilitate understanding of the data. Moreover, the abstraction may distort the information to achieve a simplified and interpretable view of the data.

There are different levels of abstractions from very-abstract views to lesser abstract views of the data. There are different types of abstractions including those that reduce the level of detail [8], throw away information, augment and adjust the position or value of the information and abstractions that depict information attributed and calculated from the data (including statistical and metadata).

Waltz uses the notion of groups from the generalization and specialization procedures to generate the abstractions of the data; each of the abstractions display representations of the grouped data. The Waltz abstractions include:

List of Groups Abstraction A list of the group names and respective material type is displayed.

Slice Abstraction Two dimensional slices through the data are shown.

Surface Abstraction A three dimensional surface is generated round each of the groups of data. The algorithm walks around the surface boundary of each three dimensional group, generating a connected mesh of polygons using an extended Marching Cubes Lookup table [11].

Skeleton Abstraction A skeleton of an image, like the skeleton of an animal, is generated. The algorithm is generated from a three dimensional extension to the two dimensional Sherman's algorithm (as described by [4])

Bounding region Abstraction Each of the groups are displayed using a bounding box glyph. The glyph encodes the absolute size (extent) and position of a group.

Net-display Abstraction The net-display abstraction depicts the data as a connected graph of nodes (a network). Each group is represented by a spherical node with the adjacency information encoded by connecting lines. The nodes are positioned to represent the average position of each of the elements in the group and the diameter is related to the number of elements in the group.

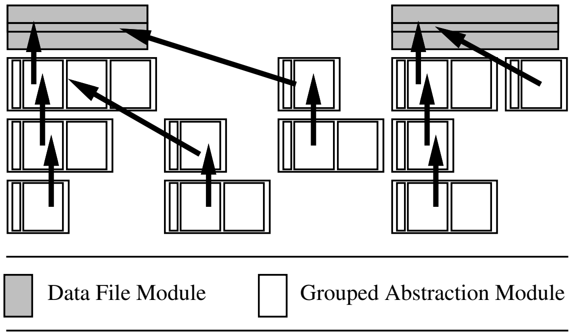

Figure 2: Waltz Right Side Rule

3.4 Waltz Data File Module

The Data File module holds the data, displays the file name and provides the root module of the specialization (tree) hierarchy. When this module is deleted each of children are likewise removed and the remaining modules are automatically re-positioned. This module is automatically generated when a Waltz Data file is loaded. The Waltz Data File is a metafile, containing the whereabouts, dimensions and aspect ratio of the data file.

3.5 Waltz Grouped Abstraction Module

Each Grouped Abstraction module contains information about the current generalization and specialization state, details of the current abstraction methods and the linkage states. Unique reference names are allocated to the abstractions to match up the abstraction popup window with the Waltz Canvas. These popup windows can be iconised into the Grouped Abstraction module, depicted by a ‘rubbed out’ bitmap representation of the Abstraction Type.

The user *Exports* the data from level to level generating multiple Grouped Abstraction Modules that are automatically positioned according to the *Right Side Rule* layout strategy.

3.6 Waltz Module Layout

Waltz automatically places the modules on the canvas, using a *Right Side Rule* method, where each first child of a parent lines up with the left side of the parent widget and each of the other children are forced to the right of that one child (Figure 2). The children of the children are placed likewise first to line up with the parent, then to display to the right of the older child and to the right of any other children (of this parent) that are older. The layout consists of multiple levels and columns. Another data set would be displayed on the right of the most right child of the previous data set.

Each Grouped Abstraction module can contain multiple abstractions (of the same or different type). The user can create and remove the abstraction methods contained within a Grouped Abstraction module, so the module automatically expands and contracts to contain the abstraction methods. The other modules in the canvas move appropriately to accommodate the expanded module.

Multiple windows are generated, containing each abstraction, and are uniquely labelled using numbers separated by dots; consisting of the Data File Version, specialization level, abstraction number and specialization path. The four-tuple label correlates each separate window with its respective Grouped Abstraction Module on the canvas. The appropriate labels are updated when an abstraction is deleted.

3.7 Waltz Linkages and Control

Views that are depicted at a high level of abstraction are often difficult to interpret, therefore, we claim there is a need to join or link these abstract views together to disambiguate the abstractness of the image. An analogy is taken from the London U.K. Underground Railway network map. The map is drawn in an abstract form and displays the connectivity of the railway stations and lines, but loses information about the exact position of a station and the distances between each station. The user can quickly understand the connectivity of the stations and effectively navigate the network. However, when the abstract map is used in isolation, it is difficult to evaluate whether it is feasible to walk overground between two stations. This distance information is retrieved by consulting a geographical map of the same area; the stations represent the landmarks of correlation and the distance evaluation becomes trivial – the *lost* information has been regained.

We have discovered six close coupling methods that are applicable to the Waltz visualization system. Each of the linkages can be applied to a single abstraction (Self Linked) or connected across multiple views (Local Linked).

Linked Highlight This allows the same data elements to be highlighted between views. Multiple groups are highlighted in different colors.

Linked Specialization Grouping A specialization is created when parts of the (generalized) data are extracted. The groups are extracted when the user initiates an *Export* action generating a specialization (subset) of the groups that are currently Highlighted. If no groups are highlighted a complete copy of the current groups is exported. The *Export* request either updates a current Grouped Abstraction Module or creates a new Grouped Abstraction Module with the new data specialization.

Linked Global Transform Direct Manipulation of rotation, scale and translation in one three dimensional view controls the orientation and view of other connected three dimensional abstractions.

Linked Group Transform This allows a transformation to be allocated to a specific group, corresponding views (within a generalization), having the same transformation applied. This is useful if a group is obscuring another group, or if a group needs to be moved spatially away from surrounding elements.

Linked Attributes The color, transparency, texture and other attributes are uniform between linked abstractions. The user can alter the displayed attributes for any group.

Linked Data Probes The position of data probes (the displayed slice in the slice abstraction, for example) are linked between views.

Within a Grouped Abstraction Module each of the abstractions are initially linked together. A button on the Grouped Abstraction module generates a popup 'form' to change the state of the Linkages. The form allows each linkage type to be switched from Local (the default state) to Self. Self linking guarantees that alterations to the abstraction affect only itself; local linking provides a mechanism to 'side effect' any other Linked Abstraction (within one Grouped Abstraction module). The 'form' (Figure 3) consists of five multiple radio-buttons that can switch state between Local and Self. The state of each Linkage is abbreviated into a Link Status symbol, with the 'Self' Linkages circled. There are only five linkages because Linked Specialization uses the local values from Linked Highlight.

Global Linkage would also be possible, but has not been implemented in this version of Waltz. Lines or pipes would be drawn on the canvas to note the specific connections with connections to any Grouped Abstraction Module. This would be useful to link abstract views between different generalizations and other data sets.

Alternating the Linkage States (in the Linkages Form) automatically updates appropriate view information in the abstraction. When an operation, in one display, generates a linkage request (such as Highlight) the linked displays are automatically updated.

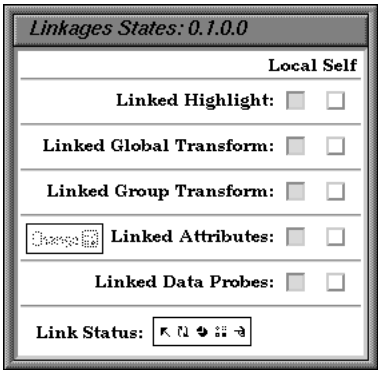

Figure 3: Waltz Link Form

3.8 Waltz Implementation

The name Waltz is *coined* from the music term of three parts in four, from the *Generalization*, *Specialization* and *Abstraction* processes within a graphical user-interface.

The user controls Waltz to generalize the data into groups; where each group is allocated a unique numerical identifier. The user interface is simplified by providing two modules (the Data File and Grouped Abstraction modules) where the user adds (and deletes) abstraction and grouping methods to the Grouped Abstraction Module, to control the visualization.

Waltz has been designed in C++ with classes representing the data, groupings of the data and abstractions, and is implemented over X and Xlib using the Motif widget set, Inventor [33, 28, 24] and Open GL [15] libraries.

4 Related Work

Abstractions if viewed on their own depict part of the information, but like the analogy of the underground map other views need to be consulted. Therefore, *linking* mechanisms between the views are useful and imperative if the *lost* information is to be retrieved. Waltz provides six linking mechanisms or *Linkages* (see section 3.7, page 4).

Many systems allow the direct manipulation of the output and linking between views. Martin and Ward provide five selection or brushing techniques in their XmdvTool [13] of: Highlight (making some points stand out), Linking (causing data points to be selected between displays), Masking/deleting (causing some points to not be displayed), Moving average (to show the average values of the selected points) and Quantitative presentation (displaying the actual data values).

Klinker [10] describes a method using "Data Probing" allowing the data to be graphically filtered by defining polygon areas of interest on the screen, using a "Cursor Linking" technique that allows the probe information to be linked between modules (and views).

The Visualization Input Pipeline (VIP) [6] provides a backward control path (from the image to the data) used to transport control information backwards up the flow pipeline. In one example, the position of the cursors are linked together and the translation of one pointer moves the position of each corresponding cursor.

The Waltz visualization system simplifies the data by generalizing the data into groups, a subset of these groups can be selected for display and further processing. Schroeder, Lorensen, Montanaro and Volpe [23] use a similar technique, named the "Display filter", allowing the output from one display to be input to another display.

Walsum and Post [30] describe a method that simplifies the data by using selection criteria. There are three stages: *selection creation* to feature extract; *selection processing* applying a transformation to the selected data; and *selective visualization* that displays the data. They state that a selection is an area of interest to the user and can be created by both spatially-connected (as of Klinker [10]) and spatially-unconnected (as of Schroeder, Lorensen, Montanaro and Volpe [23]). Their selections are boolean arrays of data. They describe algorithms of clustering, filtering the size of clusters, selection of clusters, and enlarging and reducing the size of clusters (dilation and erosion, respectively) that can be applied to create and alter selections.

Silver [26] segments data into groups named Objects. The objects simplify the data and can be tracked as they evolve and move throughout the data [22].

The Abstractions within Waltz represent simplified or transformed views of the data. Many multidimensional visualization tools provide abstract representations of the data [32, 31]. Icons or glyphs provide a high level of abstraction and data reduction and that the data reduction is "... necessary for understanding the information inherent in very large data sets" [17]. The Tioga system [27] generates child abstractions (representing subsets of the data input) on a query to the database. The children at each level are viewed and controlled by browsers. The browsers can also be synchronized, so any manipulation on one browser (either slave or master) controls a corresponding translation on the other. Program visualization systems also use abstract methods to visualize and display working models of algorithms and techniques [21].

5 Examples

The system has been used on a number of data sets. We describe a session visualizing the data from a space dust impact simulation (Figure 4). A stationary block of material is *bombarded* with a smaller piece of material. The data set describes the pressure of the impact on the object; positive (expansion) and negative (compression) pressures are represented by the data, and the air is given a reference value. The data is scaled appropriately so the most negative pressure is at the value one and the reference value is set at zero. The data is loaded into Waltz and two paths are created. The first path selects two groups to visualize the surround of object block. The second path provides multiple views displaying different generalizations of the pressure inside the stationary block.

One of the data groups (shown in Figure 5) has been *pulled off* the remaining data. An Inventor manipulator surrounds the group and the other connected views are automatically updated. The top-left part (of Figure 5) displays a view with the 'Group Transform' and 'Attribute Linkages' un-linked from the other views showing the same data with the outer group semi-transparent.

The second example segments the brain in a reduced version of the MRI Head data set: due to memory limitations (Figure 6). Firstly, the data is partitioned into two groups and the group representing the head is exported and further generalized into a number of groups. The connectivity of the groups are shown in the net-display abstraction and is used to move the brain segment away from the other tissues. The second view displays the same data with the transparency attribute value of the surface groups increased.

This section provided a brief introduction to the visualization environment; we refer the reader to the 'Quick Start' [19] and 'User Manual' [20] documents for more information about 'how to use' the environment.

6 Discussion

The Waltz system has been evaluated by a small number of volunteers. The multiple views with implicit linkages were found to be useful and the Highlight (Specialization), Global Transform and Group Transform Linkages were the most frequently used. However, other 'user defined' abstract views were required.

Waltz could be enhanced to include: other abstract methods, different grouping and segmentation techniques (such as a flood fill threshold grouper), and abstract linkages between any Grouped Abstraction Module or specific abstraction.

The generalization stage is often processor intensive; data coherence could be used to speed up this segmentation phase.

The specialization process generates a fan-out hierarchy; a merge module would provide a mechanism to merge two or more data specializations into one. Obviously, contentions of overlapping data points and registration of the data would need to be addressed. Waltz could also be extended to operate on data of different dimensions (like [32]).

We propose to develop and extend the Waltz metaphor and to provide a framework that is user extensible; allowing the user to incorporate abstraction methods into the system.

7 Conclusions

We have presented a visualization system that displays data in different linked abstract forms. The system provides a new metaphor allowing a user to segment the data into similar regions (generalization), reduce the amount of data (specialization) and display the data using a number of connected views at different levels of detail via several methods (abstractions).

Some methods of visualization are more appropriate than others to display a particular data style. The abstraction system would be useful if the data is large, complex and can be viewed in a variety of methods. Three (and higher) dimensional data sets would take advantage of the inherent linked abstraction technique (like [13, 31]).

The multiple linked views help disambiguate the abstract views: with one abstraction controlling another view. Some of the abstractions that provide insight into the data are useful for navigation and others provide mechanisms of control into related abstractions. These 'control abstractions' (list abstraction, for example) are often individually meaningless and must be used in conjunction with other abstractions to locate, select, highlight and transform regions of interest on other abstractions.

The system simplifies the visualization process to a series of specializations from generalized data, a Data Module and an Abstraction Module with multiple implicitly linked abstraction methods.

8 Acknowledgements

I acknowledge Steve Hill for his advice on this project, the Kent Physics Laboratory, U.K. for the Impact Data, the Chapel Hill site for the Head data and my parents for their encouragement and support.

References

[1] Greg Abram and Lloyd Treinish. An extended data-flow architecture for data analysis and visualization. In *Proceedings Visualization '95 – sponsored by the IEEE Computer Society*, pages 263–270, 1995.

[2] Jacques Bertin. *Semiology of Graphics, translation from Sémilogie graphique (1967)*. The University of Winsonsin Press, 1983. William J. Berg (Translator).

[3] D. S. Dyer. A dataflow toolkit for visualization. *IEEE Computer Graphics and Applications*, 10(4):60–69, 1990.

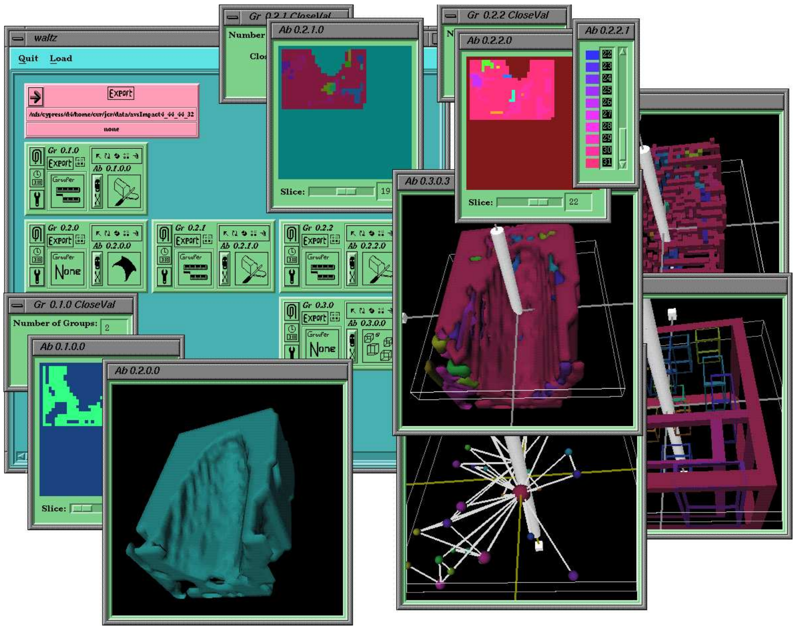

Figure 4: Waltz Canvas – Space Impact Simulation

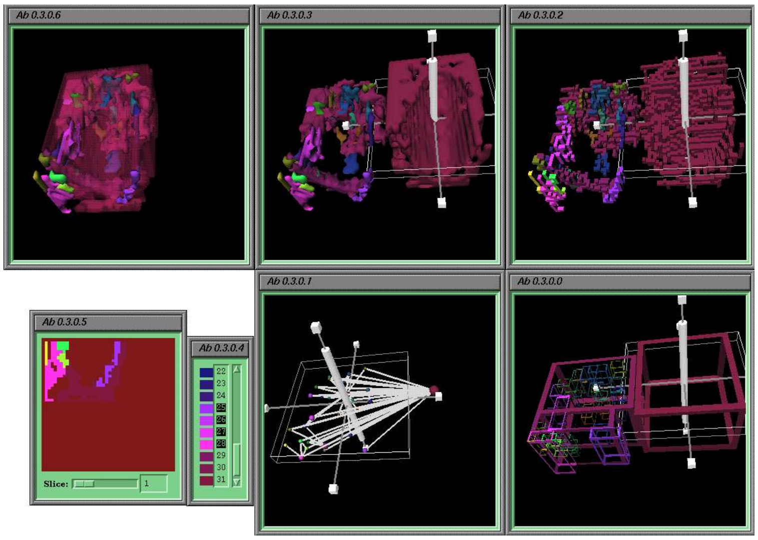

Figure 5: Abstractions showing Linkages

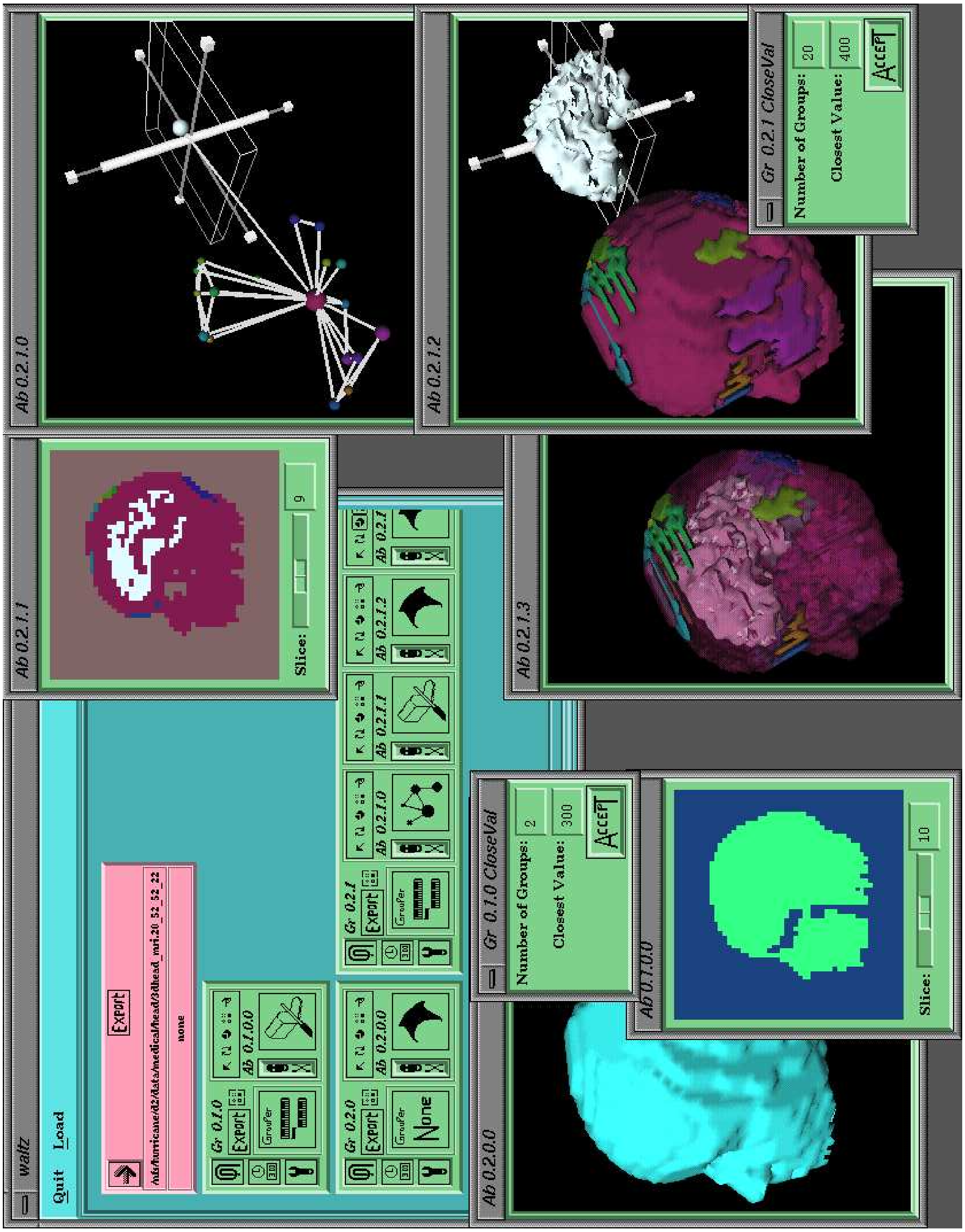

Figure 6: MRI Head Scan

[4] Michael C. Fairhurst. *Computer Vision for Robotic Systems – An Introduction*. Prentice Hall, 1988.

[5] Jean M. Favre and James Hahn. An object oriented design for the visualization of multi-variable data objects. In *Proceedings Visualization '94 – sponsored by the IEEE Computer Society*, pages 318–325, 1994.

[6] W. Felger and F. Schröder. The visualization input pipeline – enabling semantic interaction in scientific visualization. In *Eurographics '92 (Computer Graphics Forum Volume 11 No. 3) – Alistair Kilgour and Lars Kjelldahl Eds.*, pages 139–151. Blackwell Publishers, 1992.

[7] R. B. Haber and D. A. McNabb. Visualization idioms: A conceptual model for scientific visualization systems. In B. Shriver, G. M. Nielson, and L. J. Rosenblum, editors, *Visualization in Scientific Computing*, pages 74–93. IEEE Computer Society Press, 1990.

[8] T. He, L. Hong, A. Kaufman, A. Varshney, and S. Wang. Voxel based object simplification. In *Proceedings Visualization '95 – sponsored by the IEEE Computer Society*, pages 296–303, 1995.

[9] J. P. M. Hultquist and E. L. Raible. SuperGlue: A programming environment for scientific visualization. In *Proceedings Visualization '92 – sponsored by the IEEE Computer Society*, pages 243–250, 1992.

[10] Gudrun J. Klinker. An environment for telecollaborative data exploration. In *Proceedings Visualization '93 – sponsored by the IEEE Computer Society*, pages 110–117, 1993.

[11] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM Computer Graphics*, 21(4):163–169, July 1987.

[12] G. Manos, A. Y. Cairns, I. W. Ricketts, and D. Simclair. Automatic segmentation of hand-wrist radiographs. *Image and Vision Computing*, 11(2):100–111, 1993.

[13] Allen R. Martin and Matthew O. Ward. High dimensional brushing for interactive exploration of multivariate data. In *Proceedings Visualization '95 – sponsored by the IEEE Computer Society*, pages 271–278, 1995.

[14] James L. Montine. A procedural interface for volume rendering. In *Proceedings Visualization '90 – sponsored by the IEEE Computer Society*, pages 36–44, 1990. (The Alliant Volume Visualization Environment – AVOLVE).

[15] Jackie Neider, Tom Davis, and Mason Woo. *OpenGL Programming Guide – The Official Guide to Learning OpenGL, Release 1*. Addison-Wesley, 1994.

[16] Thomas C. Palmer. A language for molecular visualization. *IEEE Computer Graphics and Applications*, 12(2):23–32, May 1992.

[17] Frank J. Post, Theo van Walsum, Frits H. Post, and Deborah Silver. Iconic techniques for feature visualization. In *Proceedings Visualization '95 – sponsored by the IEEE Computer Society*, pages 288–295, 1995.

[18] Jonathan C. Roberts. *Aspects of Abstraction in Scientific Visualization*. Ph.D thesis, Kent University, Computing Laboratory, Canterbury, Kent, England, UK, CT2 7NF, October 1995.

[19] Jonathan C. Roberts. Waltz Quick Start. Technical Report 23-96, Computing Laboratory, University of Kent, Canterbury, UK, December 1996.

[20] Jonathan C. Roberts. Waltz User Manual. Technical Report 22-96, Computing Laboratory, University of Kent, Canterbury, UK, December 1996.

[21] Gruia-Catalin Roman and Kenneth C. Cox. A taxonomy of program visualization systems. *IEEE Computer*, 26(12):11–24, 1993.

[22] Ravi Samtaney, Deborah Silver, Norman Zabusky, and Jim Cao. Visualizing features and tracking their evolution. *IEEE Computer*, 27(7):20–27, July 1994.

[23] W. J. Schroeder, W. E. Lorensen, G. D. Montanaro, and C. R. Volpe. VISAGE: an object-oriented scientific visualization system. In *Proceedings Visualization '92 – sponsored by the IEEE Computer Society*, pages 219–226, 1992.

[24] Raj Shekhar and Brian McGinley. Open Inventor 2.0. *Computer – IEEE Society Press*, 27(7):100–102, July 1994.

[25] Silicon Graphics Computer Systems – Silicon Graphics Inc. *IRIS Explorer Technical Overview*, April 1992. (http://www.nag.co.uk/visual/IE/iecbb/docs/Technical_Report.ps).

[26] Deborah Silver. Object-oriented visualization. *IEEE Computer Graphics and Applications*, 15(3):54–62, May 1995.

[27] Michael Stonebraker, Jolly Chen, Nobuko Nathan, Caroline Paxon, Alan Su, and Jiang Wu. Tioga: A database-oriented visualization tool. In *Proceedings Visualization '93 – sponsored by the IEEE Computer Society*, pages 86–93, 1993.

[28] Paul S. Strauss and Rikk Carey. An object-oriented 3D graphics toolkit. *Computer Graphics*, 26(2):341–349, July 1992.

[29] C. Upson, T. Faulhaber, D. Kamins, D. Schlegel, D. Laidlaw, F. Vroom, R. Gurwitz, and A. van Dam. The application visualization system: A computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, 9(4):30–42, 1989.

[30] Theo van Walsum and Frits H. Post. Selective visualization of vector fields. In *Eurographics '94 – M. Daehlem and L. Kjelldahl Eds.*, pages 339–347, 1994.

[31] Jarke J. van Wijk and Robert van Liere. HyperSlice – visualization of scalar functions of many variables. In *Proceedings Visualization '93 – sponsored by the IEEE Computer Society*, pages 119–125, 1993.

[32] Matthew O. Ward. XmdvTool: Integrating multiple methods for visualizing multivariate data. In *Proceedings Visualization '94 – sponsored by the IEEE Computer Society*, pages 326–333, 1994.

[33] Josie Wernecke. *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Release 2*. Addison-Wesley, 1994.

[34] Mark Young, Danielle Argiro, and Steven Kubica. Cantata: Visual programming environment for the Khoros system. *Computer Graphics*, 29(2):22–24, May 1995.