Visualizing the World-Wide Web with the Navigational View Builder

Sougata Mukherjea, James D. Foley
Graphics, Visualization & Usability Center
College of Computing
Georgia Institute of Technology
E-mail: sougata@cc.gatech.edu foley@cc.gatech.edu

Abstract

Overview diagrams are one of the best tools for orientation and navigation in hypermedia systems. However, constructing effective overview diagrams is a challenging task. This paper describes the Navigational View Builder, a tool which allows the user to interactively create useful visualizations of the information space. It uses four strategies to form effective views. These are binding, clustering, filtering and hierarchization. These strategies use a combination of structural and content analysis of the underlying space for forming the visualizations. This paper discusses these strategies and shows how they can be applied for forming visualizations for the World-Wide Web.

KEYWORDS: Information Visualization, Overview Diagrams, Binding, Clustering, Filtering, Hierarchization.

1 Introduction

One of the major problems with current hypermedia systems is being *lost in hyperspace*. For example in Mosaic [1] the process of jumping from one location to another can easily confuse the user. This is primarily a result of the user's lack of knowledge of the overall structure of the information space. For this purpose overview diagrams or *navigational views* are very useful. By presenting a map of the underlying information space, they allow the users to see where they are, what other information is available and how to access the other information. In fact, they are one of the best tools for orientation and navigation in hypermedia documents [12].
However, construction of effective overview diagrams is a challenging task. There are various problems involved:

- The navigational views are two or three dimensional projections of generally multidimensional hypermedia networks. Finding effective views of such complex network structures that convey all the required information is hard. In particular, this task is impossible without knowledge of the structure and the contents of the information space.

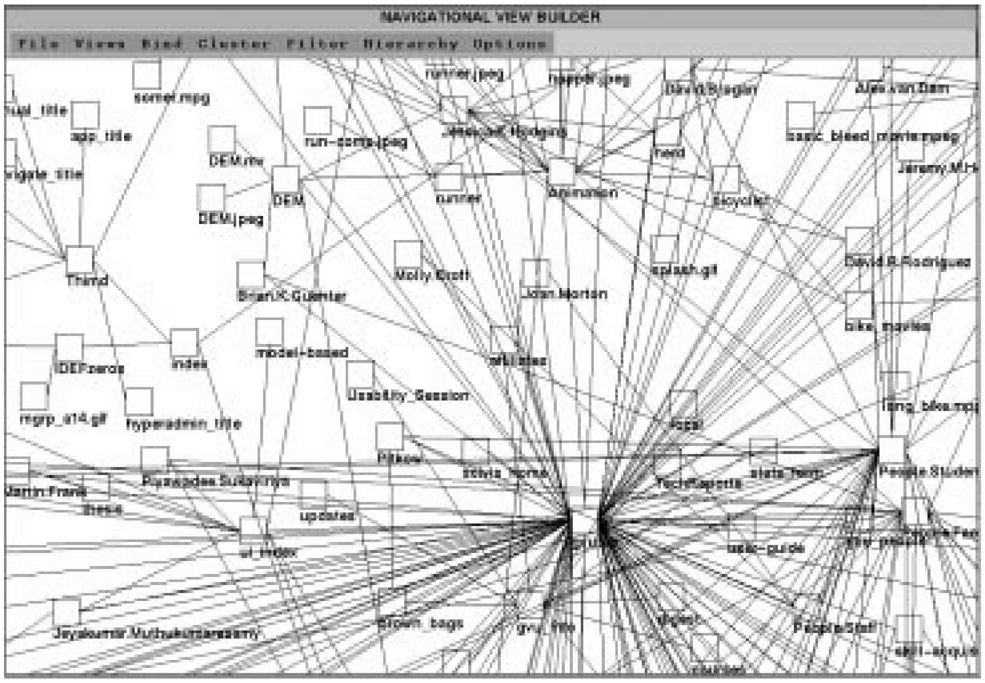


Figure 1: An overview diagram of the World-Wide Web pages about the research activities at GVU. It indicates clearly why traditional overview diagrams are useless for real-world hypermedia systems.

- Even if such a view is developed, the resulting structure would be very complex for any non-trivial hypermedia system. Creating an aesthetic layout of such a complex structure is extremely difficult. In fact the overview diagrams are generally represented as node and link graph diagrams and graph layout is a very complex problem [2].
- As the size of the underlying information space increases, it becomes very difficult to fit the whole information structure on a screen. If the size is reduced to fit on the screen, the details become too small to be seen. An alternative is to browse the large layout by scrolling and arc traversing. However, this tends to obscure the global structure. The goal should be to display both the details and the context smoothly integrated together in a single screen.
- Just displaying the structure to the user is not enough. To be really useful, the user should be able to get an idea of not only the structure but the actual contents of the nodes and links just by looking at the navigational views.

We are building the **Navigational View Builder**, a tool for letting the designer develop effective overview diagrams of hypermedia systems. (The implementation is done using C++, Motif and Open Inventor [13]). The tool uses various strategies to reduce the problems concerned with developing overview diagrams. This paper discusses these strategies and shows how they can be used to form useful views of the World-Wide Web.

We assume a database-oriented hypermedia system where the nodes are described with attributes. Since the WWW is unstructured and does not have this model of the hypermedia system, the model was built from the node and link structure of the WWW extracted by parsing the html documents using the strategy described in [10]. Some of the attributes of the nodes like the author (the owner of the file) and the file-size could be extracted automatically from the files. However, a major drawback of the World-Wide Web is the absence of many useful semantic attributes for the pages. Therefore, to fully show the power of our tool, attributes like the topic of the page (whether it is a research page or a personal page, etc.) were inserted manually. (Efforts are underway to incorporate metadata into WWW and hopefully in the near future we can extract all useful information from WWW automatically).

Figure 1 shows an unstructured overview diagram of the WWW pages about the research activities at the Graphics Visualization & Usability (GVU) Center at Georgia Tech.[1] These pages mainly contain information about the faculty, students and research activities of GVU. This figure shows how useless such overview diagrams can be for helping the user understand anything. The whole information space does not fit on a screen, and the interconnections between the nodes make the structure too complex for easy understanding. The figure also does not give any details of the characteristics of the nodes and links. The following sections will describe various strategies to make this diagram more understandable.

2 Binding

To make the overview diagrams useful for the user, they should depict more information than merely the structure of the information space. The user should get an idea of the content of the space so that just by looking at the diagram the user can decide what part of the overall information space is of interest to her. For example, in the WWW, an user should be able to get an idea about the topic of the page by just looking at the overview diagram. Similarly, if certain nodes are important (known as *landmarks* in hypermedia literature), they should be identifiable from the views. To achieve this purpose, visual properties can be used in the overview diagram to represent information from the underlying information space. For example, the topic of the nodes may be represented by different colors. Similarly, special icons or brighter colors can be used to represent the landmark nodes.

The designer of the overview diagrams should be able to specify the *bindings* between the information attributes and the visual attributes of the nodes and links. These bindings specify a mapping between the information space and the visual space of the hypermedia system. At the time of displaying the navigational views, these bindings and the information in the underlying information space control the actual appearance of the views. For example, if the topic is bound to the color of the nodes, the color of the nodes in the view would be controlled by the topic of the page which the node represents. Unfortunately, the number of visual properties that can be used to depict information is usually less than the amount of information in the underlying database. Therefore, only the important data attributes should be bound to the visual properties. Thus, the mapping between the database and the visual attributes should not be hardwired into the system since the importance of the data attributes cannot be fixed a-priori. The Navigational View Builder allows the designer of the navigational views to specify the *bindings* between the data and the visual attributes of the nodes and links in an easy-to-use interface. A sample interface for binding node attributes is shown in Figure 2. Figure 3 shows an example of a view of the GVU WWW pages with various visual properties bound to information attributes. In this example icons are used to represent various media types, the node size is bound to the file size (inversely) while the shape represents whether the author of the page was a student (circles represent students). The color hue represents the topic of the page while the color saturation represents the last modified time.

[1]URL: http://www.gatech.edu/gvu/gvutop.html

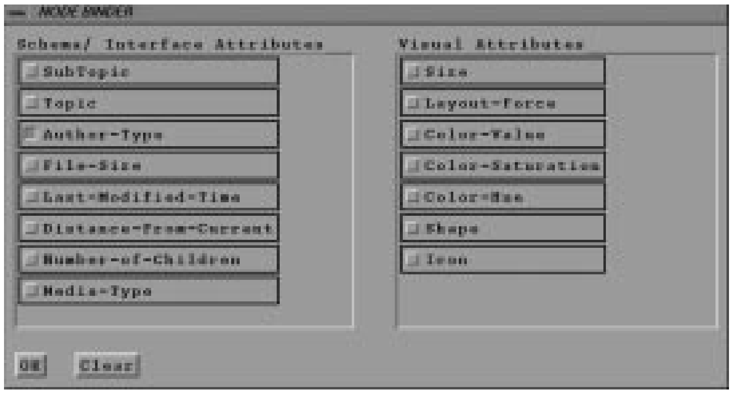


Figure 2: Piece of the interface for binding visual attributes to database attributes.

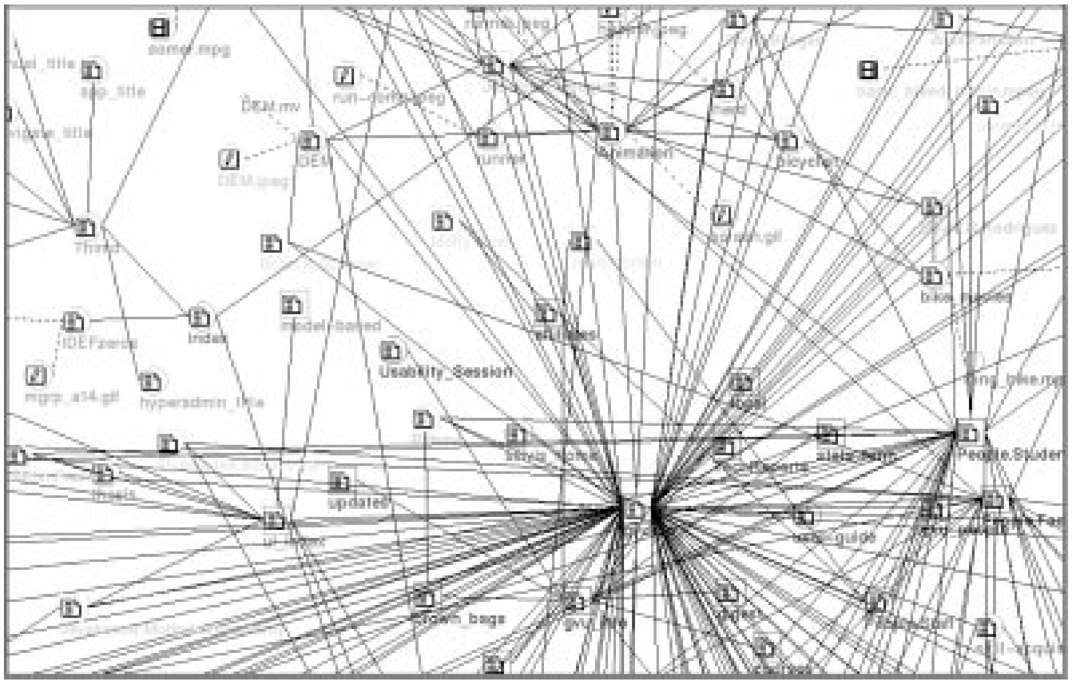


Figure 3: An example of binding for the WWW. Visual attributes are used to represent information.

Moreover, links connecting two html files are solid while links connecting html files to other kind of files are dashed. More information is presented to the user in this diagram than in Figure 1. A more detailed discussion of our binding strategy and its usefulness is presented in [5].

3 Clustering

One of the major problems with the overview diagrams is that the screen gets cluttered with too much information for even a reasonably sized hypermedia system. One way to overcome the problem is to reduce the information that needs to be shown on the screen through suitable abstractions. These abstractions will show the user the overall information space without cluttering the screen. For example, it is obviously not possible to show the entire GVU WWW pages on a single screen. Hence we may form an abstraction which clusters all files in the same directory into a single node.

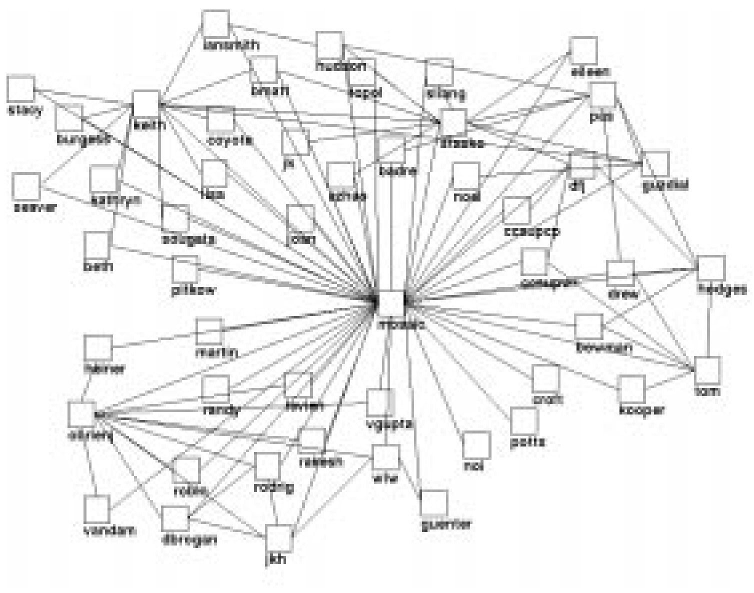


Figure 4: An example of similarity-based clustering: Clustering done by authors. Clusters for the different authors of the GVU WWW pages are formed. Shapes represent the type of the authors (circles represent students).

Clustering is possible for the abstracted views also. For example, if the number of directories are large, even the abstraction which represents each directory as a single node may be too confusing for the user. Hence there may be further clustering based on, for example, the topic of the pages. These abstracted views would be able to show the overall information space on a single screen without much cluttering. Hence the user would have a better feel of the global information structure from these abstracted views. Depending on the intentions and goals of the users, different kinds of abstractions would be useful to them. For example, one user may want to see an abstraction of the information space based on the topic. Some other user may want to see the abstraction based on the author of the pages. Hence, for the abstracted navigational views to be really useful, we should allow the interactive specification of the abstractions.

3.1 Types of Clustering

Two types of clustering are useful. They are as follows:

1. **Structure-based clustering:**
   In structure-based clustering, the hypermedia structure is searched for subnetworks that match a given pattern. These subnetworks form a cluster. The simplest form of this type of clustering is **link-based clustering** in which nodes that are joined by a particular class of links are clustered together. For example, in an overview diagram we may want to form clusters of nodes that are linked by *annotation* links.

2. **Content-based clustering:**
   In this type of clustering the nodes of the hypermedia are considered individually and their attributes are examined to determine the clusters. The simplest form of content-based clustering is when all nodes whose attributes satisfy certain properties form a cluster. Sometimes

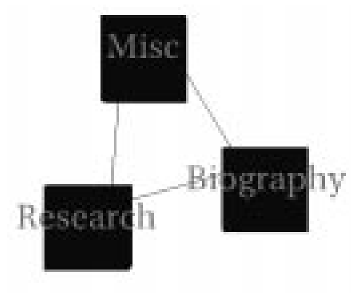


Figure 5: An abstracted view of the GVU's WWW research pages. Succesive abstractions done by directory, sub-topic, topic, etc.

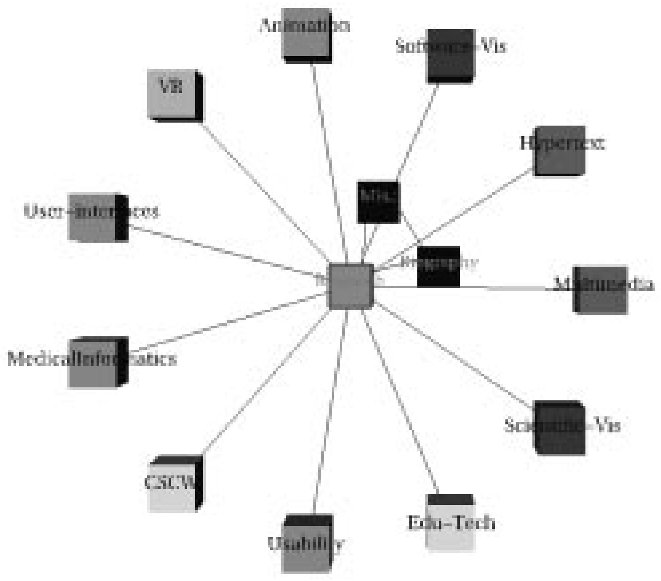


Figure 6: Viewing the abstracted structure. The user is seeing the details of Research pages. The details of each layer are shown in the x-y plane. The layers are arranged in the z dimension with the most detailed view (of the Research pages) in front.

the user would like to cluster all nodes that have the same value for a certain attribute. If the user has to generate a clustering command for each value of the attribute it would be quite tedious. Therefore, **Attribute-similarity-based** clustering is allowed in which for each value of the specified attribute, clusters of nodes having that value is formed. For example, all pages built by the same author can be clustered together (as shown in Figure 4).

To allow the clustering to be done interactively, it is essential that the clustering algorithms are efficient. We discuss our clustering algorithms in [6]. Note that clustering can be combined with binding so that information about the clusters are shown visually. Thus in Figure 4, shapes represent the type of the authors (circles represent students).

3.2 Visualizing the Abstracted Views

By a series of clustering, the user will be able to see the global view of the information space. After the user gets an idea of the global structure, she would generally want details of some part

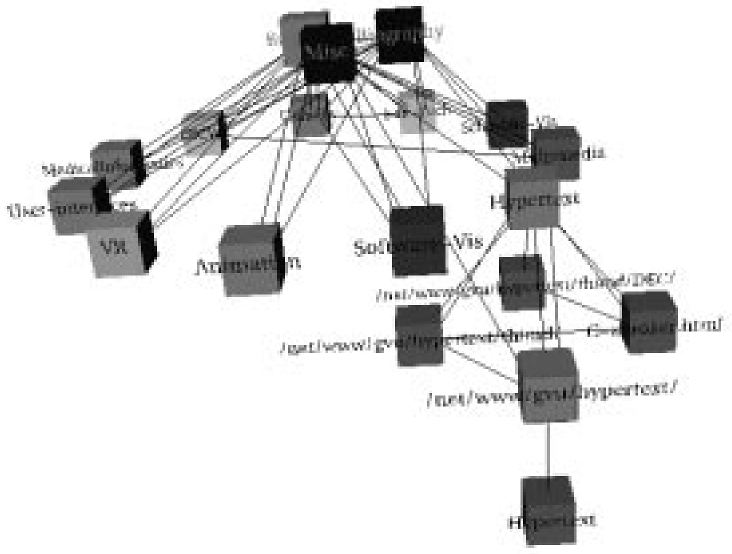


Figure 7: A top view of the structure making the hierarchy formed by the abstraction layers apparent. The user is seeing the details of the Hypertext research area. Colors represent different research areas.

of the space and the problem with the abstracted views is that the details would be lost to the user. To show the details together with the context, we have developed a visualization strategy using 3 dimensions. Initially the user will be shown the most abstracted view as in Figure 5. (Here the abstraction was done by directory, sub-topic, topic, etc.) When the user wants the details of a particular node, those details will be shown at the front while the more abstract layers move deeper. For example, Figure 6 shows a view where the user wanted to see the details of the Research pages. The x-y plane is used for showing the details of each particular layer and the layers are arranged in the z dimension with the most detailed view (of the Research pages) in the front. It can be argued that as more and more details are available, the context would move deeper and deeper into the screen. However, the eye-point can be shifted to bring different parts of the space in focus. For example, besides the front view, back, side and top views can also be provided. Figure 7 shows a top view of the information space after the user wanted to see details of the Hypertext research area. (Note that colors are used to represent different research areas). We allow smooth animation so that the view changes are not abrupt and allows the user to see the changes easily. This view shows that the details of the user's interest as well as the context are integrated together on a single screen. (However, it should be noted that the user can generally see the details of one part of the space only - otherwise the view becomes very confusing).

4 Filtering

To reduce the complexity of the overview diagrams, the user may want to remove unwanted information (or show the useful information only). Therefore, we allow various filtering options.

1. **Content-based:** In this type of filtering only nodes whose attributes satisfy certain properties are shown/ hidden. For example, Figure 8 shows only the pages related to research in

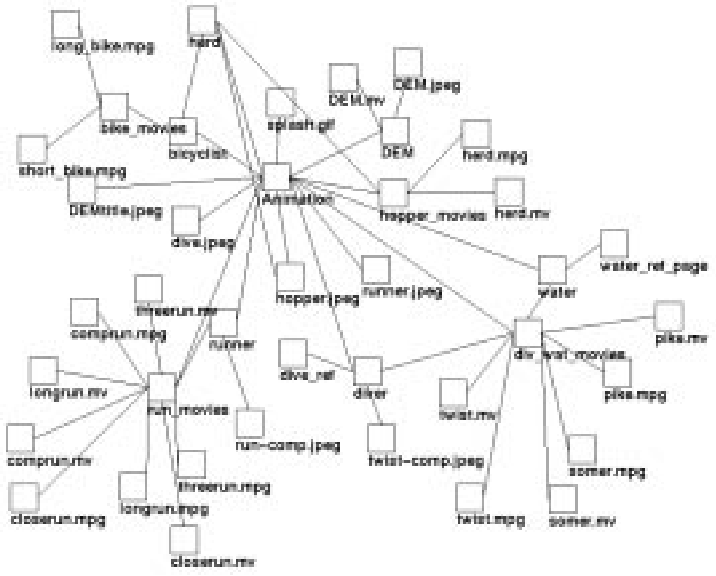


Figure 8: An example of content-based filtering: Showing only the pages related to Animation research.

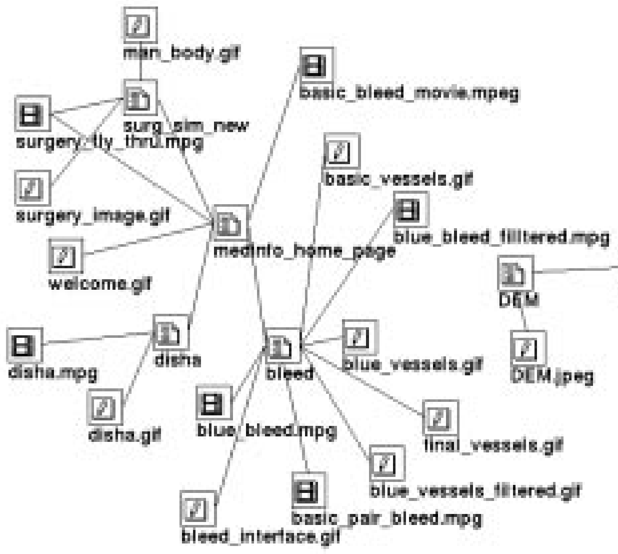


Figure 9: An example of structure-based filtering: Showing only the html pages linked to both images and movies. Icons represent the media types of the pages.

Animation.

2. **Link-based:** In this type of filtering only certain types of links are shown/ hidden.

3. **Structure-based:** Suppose some organizations have guidelines for their employees when they are making their WWW home pages. It may state that a central html page should be linked to a gif file and another html page. Later on, an administrator for these pages may wish to check whether people have been following the guidelines. Traditional content-based database querying would not be of much help. Therefore we allow structure-based querying which allows the showing/hiding of certain subgraph patterns only. For example, Figure 9 shows the html pages that are linked to both images and movies. (Icons represent the media types of the pages).

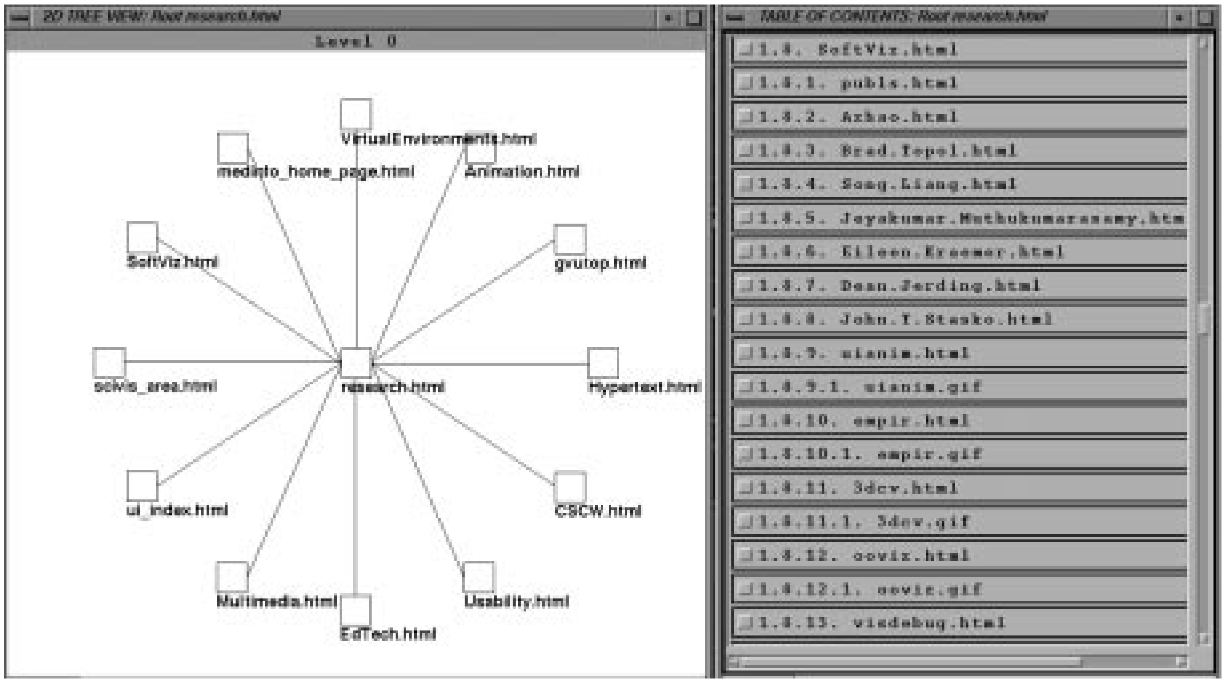
Note that our filtering algorithms are similar to the clustering algorithms. Also note that our content-based filtering is similar to the concept of dynamic queries [14].

5 Hierarchization

The previous sections have assumed that the overview diagrams would be presented as node and link graph diagrams. Since the underlying hypermedia structure is a network, this may seem to be the best way to present the information. However, graphs are the most difficult data organizations to visualize. Moreover, views of the full graph structure are difficult for the users to comprehend. Although clustering and filtering help in reducing the complexity, unless the user has some idea about the underlying information space, she would not know what type of clustering or filtering would be useful for her. Therefore, some automatic mechanism which shows the user the overall information space in an easy-to-understand visualization would be required.
In [9] Parunak notes that: “The insight for hypermedia is that a hyperbase structured as a set of distinguishable hierarchies will offer navigational and other cognitive benefits that an equally complex system of undifferentiated links does not, even if the union of all the hierarchies is not itself hierarchical.” [8] observed that the ability to view knowledge from different perspectives is important. Thus, if different hierarchies, each of which gives a different perspective to the underlying information can be formed, the user would be better able to comprehend the information. It should be also noted that unlike graphs some very effective ways of visualizing hierarchies have been proposed. Examples are *Treemaps* [3] and *Cone Trees* [11].
Therefore, we have developed an algorithm for forming hierarchies from hypermedia graphs. It uses both structural and content analysis to identify the hierarchy. The structural analysis looks at the structure of the graph while the content analysis looks at the contents of the nodes. These hierarchies can be visualized in various ways.

5.1 Visualizing WWW through Multiple Hierarchies

Our algorithm has been implemented in the Navigational View Builder. The left hand screen of Figure 10 shows the top level of the default hierarchy created for the data by our algorithm. The file *research.html* which lists the various research activities of GVU is the root. It has branches to the major research area as well as to *gvutop.html*, a file containing general information about GVU.



1.4. SoftViz.html
1.4.1. publs.html
1.4.2. Axhao.html
1.4.3. Brad.Topol.html
1.4.4. Song.Liang.html
1.4.5. Jeyakumar.Muthukumarasamy.htm
1.4.6. Eileen.Kraemer.html
1.4.7. Dean.Jerding.html
1.4.8. John.T.Stasko.html
1.4.9. uianim.html
1.4.9.1. uianim.gif
1.4.10. empir.html
1.4.10.1. empir.gif
1.4.11. 3dcv.html
1.4.11.1. 3dcv.gif
1.4.12. ooviz.html
1.4.12.1. ooviz.gif
1.4.13. visdebug.html

Figure 10: The left hand screen shows the top level of the default hierarchy formed for the GVU WWW pages. *research.html* is the root and the major research areas are shown. The right hand screen shows a book view of a portion of this hierarchy showing research in Software Visualization.

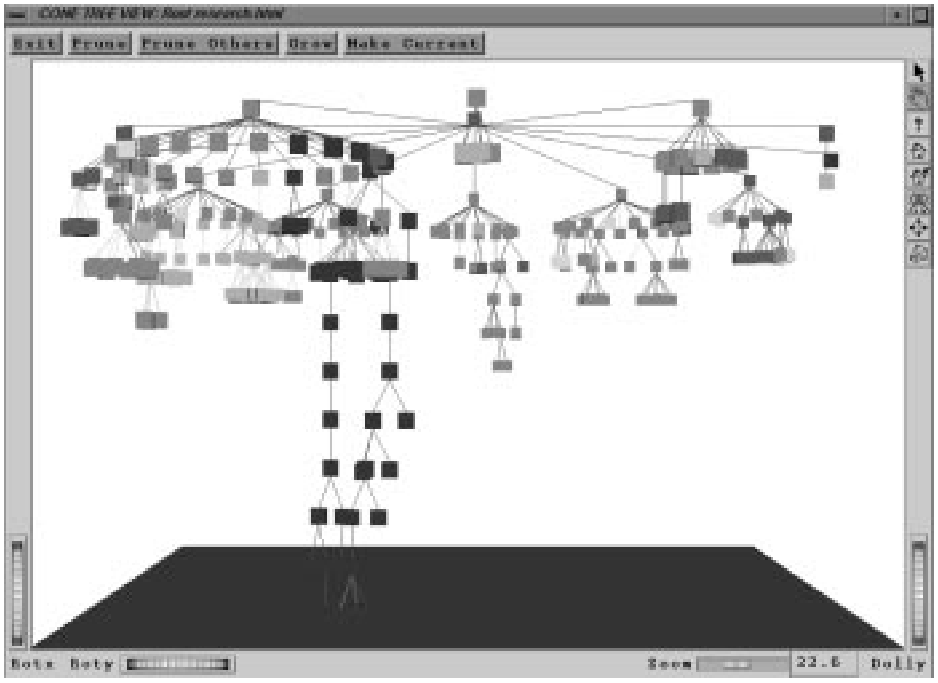
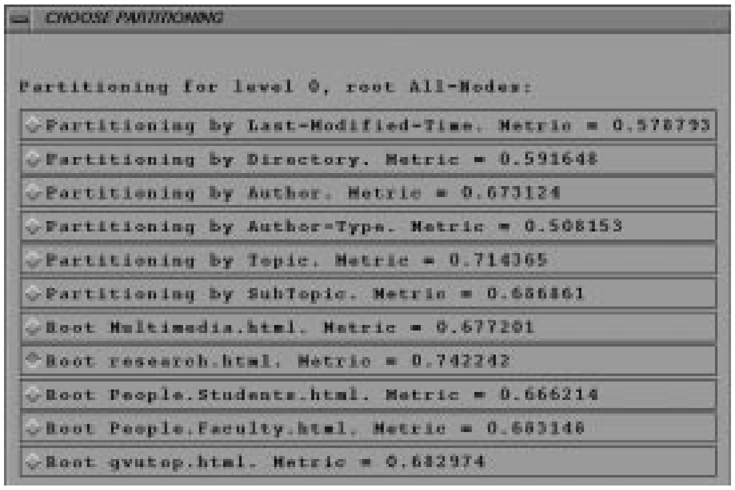


Figure 11: A 3d Tree view of the default hierarchy of the GVU research pages. The node colors represent author types while the link colors depend on the media type of the destination pages. The interface allows various zooming, rotating and filtering operations for the tree.



```
Partitioning for level 0, root All-Nodes:
Partitioning by Last-Modified-Time. Metric = 0.578793
Partitioning by Directory. Metric = 0.591648
Partitioning by Author. Metric = 0.673124
Partitioning by Author-Type. Metric = 0.508153
Partitioning by Topic. Metric = 0.714365
Partitioning by SubTopic. Metric = 0.686861
Root Multimedia.html. Metric = 0.677201
Root research.html. Metric = 0.742242
Root People.Students.html. Metric = 0.666214
Root People.Faculty.html. Metric = 0.683148
Root gvutop.html. Metric = 0.682974
```

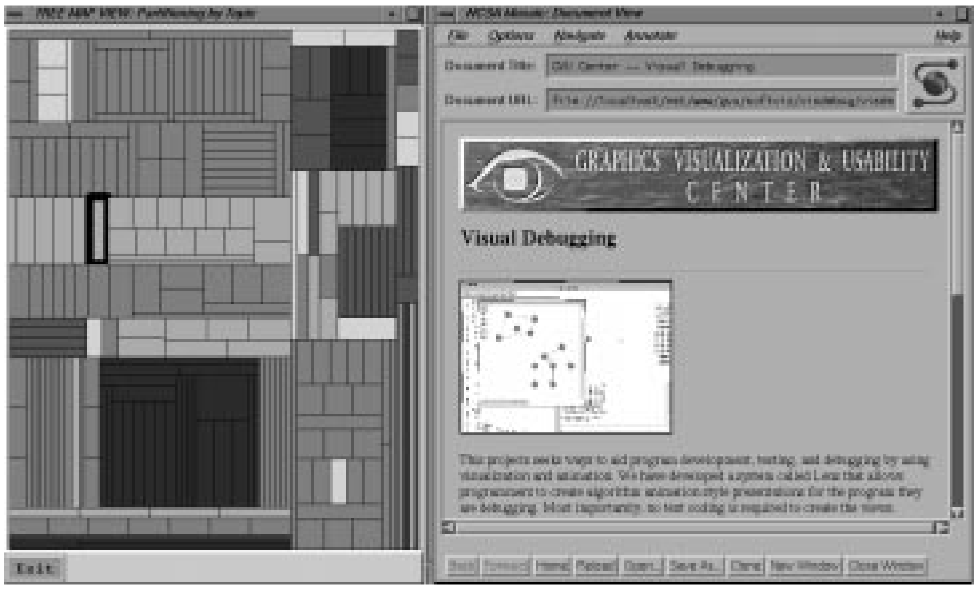
Figure 12: Our algorithm can form various hierarchies. A list of these are shown to the user ranked by a metric. The user can choose any one.

The right hand side of Figure 10 shows a view of a section of this hierarchy (showing research in Software Visualization) where the nodes are listed as a table of content of a book. Figure 11 shows a 3d tree view of this hierarchy. Binding can be used in these views as well. Thus, in this figure the node colors represent author types while the link colors depend on the media type of the destination pages. Also note that the interface allows various zooming, rotating and filtering operations for the 3d tree.

In each step, our hierarchization algorithm tries to identify a root and partition the other nodes of the graph into different branches. These branches are themselves graphs and the algorithm is recursively called for each branch. At each step various partitions are possible based on structural and content analysis. A metric is used to rank these. By default the partitioning with the best metric is chosen and thus, the algorithm forms a hierarchy automatically. However, the user can guide the hierarchization process. Various choices for forming partitions can be shown in a menu as shown in Figure 12 and the user can choose any one. Details of the algorithm is presented in [7]. The left hand screen of Figure 13 shows a Treemap view of a hierarchy that was formed with the attribute topic being used to initially partition the nodes. Here also colors are used to represent author-type. In this way, multiple hierarchies, each giving a different perspective to the underlying information space can be formed. If a user selects a node in one view, its positions in other views are also highlighted. Thus, these views help the user in comprehending the data. It should be also noted that the user can go directly to the corresponding WWW page for the selected node. Thus in the Treemap view the node *visdebug.html* is highlighted. The corresponding WWW page is shown on the right hand screen of Figure 13.

5.2 Generating Other Views

Once a hierarchy is formed from the original graph structure, other views can be developed as well. For example, if the original partitioning for forming the hierarchy was done by a quantitative attribute, a linear structure sorted by that attribute can be formed from the subtrees of the root node. For example, Figure 14 represents a perspective wall [4] view of a linear structure sorted by the last-modified-time. From the hierarchy whose initial partitioning was by the attribute last-


TREE MAP VIEW: Partitioning by topic
Exit
NCSA Mosaic: Document View
File
Options
Navigate
Annotate
Help
Document Title:
GVU Center — Visual Debugging
Document URL:
GRAPHICS VISUALIZATION & USABILITY CENTER
Back
Forward
Home
Reload
Open...
Save As...
Clone
New Window
Close Window

Visual Debugging

This projects seeks ways to aid program development, testing, and debugging by using visualization and animation. We have developed a system called Lens that allows programmers to create algorithm animation-style presentations for the program they are debugging. Most importantly, no text coding is required to create the views.

Figure 13: The left hand screen shows a Treemap view of a hierarchy of the GVU research pages where topic was used for the initial partitioning. The node colors represent author types. The node *visdebug.html* is the selected node. The corresponding WWW page is shown on the right hand screen.

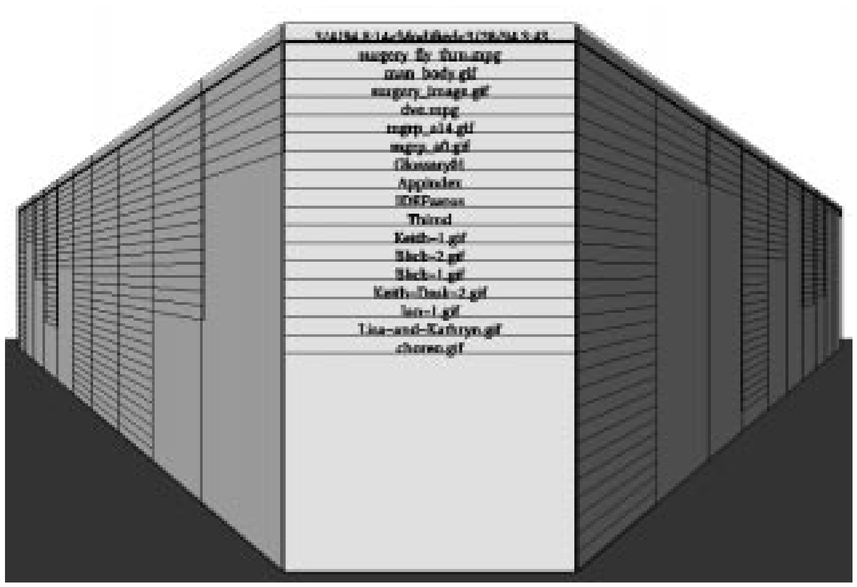


Figure 14: A Perspective Wall view showing a linear arrangement of the files based on the last modification time. The diffrent walls shows files which were last modified in different time frames. Only some walls are in the focus at a given time.

modified-time, the files were divided into partitions based on the time when they were last modified. These partitions were arranged on walls. Only some walls are in the focus at a given time. The user can easily control the walls which are in focus through a scrollbar. Similarly a tabular view showing useful statistics for the various pages and also for groups of pages organized by topic can be formed by a depth-first traversal of the hierarchical structure whose initial partitioning is done by the attribute topic.

6 Conclusion

We have presented the Navigational View Builder, a tool for forming various useful visualizations of the WWW. We believe that by using the various strategies that are described in the paper the user will be able to form visualizations that help in reducing the lost in hyperspace problem. Future work is planned along the following directions:

- A limitation of our system is that no evaluation of how useful our strategies and views really are have been done so far. We plan to do serious usability studies in the near future. These studies may give us new insights that will help to improve our system.
- At present our system runs for the research pages of GVU having about 400 nodes and 800 links. For larger data sets our hierarchization algorithm will become slower. (Presently the algorithm takes about 7 seconds on a SGI reality engine). Therefore, we are trying to use smarter data structures to improve the efficiency of the code. Moreover, for larger data sets better visualizations of hierarchies will be needed. Research needs to be done in this direction.
- To improve our content-based analysis we need to incorporate more useful metadata into our system. For example, information like the last time a file was accessed or the number of times a link was traversed would be very useful. These information can be easily accessed by incorporating a web analysis tool [10] into our system. Finding other useful metadata is an open research issue.

Acknowledgement

This work is supported by grants from Digital Equipment Corporation, Bell South Enterprises, Inc. and Emory University System of Health Care, Atlanta, Georgia as part of the Hypermedia Interface for Multimedia Databases project.

References

[1] M. Andreessen. NCSA Mosaic Technical Summary. Technical report, National Center for Supercomputing Applications, 1993.

[2] G. Battista, P. Eades, R. Tamassia, and I. Tollis. Algorithms for Drawing Graphs: an Annotated Bibliography. Technical report, Brown University, June 1993.

[3] B. Johnson and B. Shneiderman. Treemaps: A Space-filling Approach to the Visualization of Hierarchical Information. In *Proceedings of IEEE Visualization '91 Conference*, pages 284–291, San Diego, Ca, October 1991.

[4] J. D. Mackinlay, S. Card, and G. Robertson. Perspective Wall: Detail and Context Smoothly Integrated. In *Proceedings of the ACM SIGCHI '91 Conference on Human Factors in Computing Systems*, pages 173–179, New Orleans, La, April 1991.

[5] S. Mukherjea and J. Foley. Navigational View Builder: A Tool for Building Navigational Views of Information Spaces. In *ACM SIGCHI '94 Conference Companion*, pages 289–290, Boston, Ma, April 1994.

[6] S. Mukherjea, J. Foley, and S. Hudson. Interactive Clustering for Navigating in Hypermedia Systems. In *Proceedings of the ACM European Conference of Hypermedia Technology*, pages 136–144, Edinburgh, Scotland, September 1994.

[7] S. Mukherjea, J. Foley, and S. Hudson. Visualizing Complex Hypermedia Networks through Multiple Hierarchical Views. To appear in *Proceedings of ACM SIGCHI '95*, May 1995.

[8] C. Neuwirth, D. Kauffer, R. Chimera, and G. Terilyn. The Notes Program: A Hypertext Application for Writing from Source Texts. In *Proceedings of Hypertext '87 Conference*, pages 121–135, Chapel Hill, NC, November 1987.

[9] H. Parunak. Hypermedia Topologies and User Navigation. In *Proceedings of Hypertext '89 Conference*, pages 43–50, Pittsburgh, Pa, November 1989.

[10] J. Pitkow and K. Bharat. WEBVIZ: A Tool for World-Wide Web Access Log Visualization. In *Proceedings of the First International World-Wide Web Conference*, Geneva, Switzerland, May 1994.

[11] G. G. Robertson, J. D. Mackinlay, and S. Card. Cone Trees: Animated 3D Visualizations of Hierarchical Information. In *Proceedings of the ACM SIGCHI '91 Conference on Human Factors in Computing Systems*, pages 189–194, New Orleans, La, April 1991.

[12] K. Utting and N. Yankelovich. Context and Orientation in Hypermedia Networks. *ACM Transactions on Office Information Systems*, 7(1):58–84, 1989.

[13] J. Wernecke. *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor*. Addison-Wesley Publishing Company, 1994.

[14] C. Williamson and B. Shneiderman. The Dynamic HomeFinder: Evaluating Dynamic Queries in a Real-Estate Information Exploration System. In *Proceedings of the ACM SIGIR '92 Conference on Research and Development in Information Retrieval*, pages 338–346, Copenhagen, Denmark, June 1992.