Navigating Hierarchically Clustered Networks through Fisheye and Full-Zoom Methods

DOUG SCHAFFER
The University of Calgary

ZHENGPING ZUO
Simon Fraser University

SAUL GREENBERG
The University of Calgary

LYN BARTRAM and JOHN DILL
Simon Fraser University

SHELLI DUBS
Alberta Research Council

and

MARK ROSEMAN
The University of Calgary

Many information structures are represented as two-dimensional networks (connected graphs) of links and nodes. Because these networks tend to be large and quite complex, people often

Development of the variable-zoom method was part of the Intelligent Graphic Interface project, made possible through the support of Industry, Science and Technology Canada; of the British Columbia Ministry of Advanced Education, Training and Technology; and of PRECARN Associates. Research at the University of Calgary was partially funded by the National Science and Engineering Research Council.
Authors from Simon Fraser University designed and built the fisheye view system, while authors from the University of Calgary and the Alberta Research Council conducted the study and subsequent analysis. An earlier version of this work was published as Schaffer et al. [1993].
Authors' addresses: S. Greenberg, M. Roseman, and D. Schaffer, Department of Computer Science, The University of Calgary, Calgary, Alberta, Canada, T2N 1N4; email: saul@cpsc.ucalgary.ca; L. Bartram, J. Dill, and Z. Zuo, School of Engineering Science, Simon Fraser University, Burnaby, British Columbia, V5A 4S6, Canada; email: dill@cs.sfu.ca; S. Dubs, Alberta Research Council, 6815 8 Street NE, Calgary, Alberta, Canada; email: dubs@skyler.arc.ab.ca.

prefer to view part or all of the network at varying levels of detail. *Hierarchical clustering* provides a framework for viewing the network at different levels of detail by superimposing a hierarchy on it. Nodes are grouped into clusters, and clusters are themselves placed into other clusters. Users can then navigate these clusters until an appropriate level of detail is reached. This article describes an experiment comparing two methods for viewing hierarchically clustered networks. Traditional *full-zoom* techniques provide details of only the current level of the hierarchy. In contrast, *fisheye views*, generated by the "variable-zoom" algorithm described in this article, provide information about higher levels as well. Subjects using both viewing methods were given problem-solving tasks requiring them to navigate a network, in this case, a simulated telephone system, and to reroute links in it. Results suggest that the greater context provided by fisheye views significantly improved user performance. Users were quicker to complete their task and made fewer unnecessary navigational steps through the hierarchy. This validation of fisheye views is important for designers of interfaces to complicated monitoring systems, such as control rooms for supervisory control and data acquisition systems, where efficient human performance is often critical. However, control room operators remained concerned about the size and visibility tradeoffs between the fine detail provided by full-zoom techniques and the global context supplied by fisheye views. Specific interface features are required to reconcile the differences.

Categories and Subject Descriptors: H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*theory and methods*; *interaction styles*; I.3.6 [**Computer Graphics**]: Methodology and Techniques—*interaction techniques*

General Terms: Human Factors

Additional Key Words and Phrases: Data acquisition, fisheye views, hierarchically clustered graphs, information visualization, supervisory control

1. INTRODUCTION

People naturally perceive the world using both local detail and global context. While we see visual detail for only small focused regions, we retain global context through peripheral vision and by glancing around. We rely heavily on global context to orient ourselves and to understand local detail; indeed, tunnel vision is considered a serious handicap.

Unfortunately, today's computers encourage "tunnel vision" interfaces, for they supply users with very small screens to view large complex information spaces (even a 19-inch display consumes only a fraction of our normal field of view). Interface designers have developed several strategies that minimize the tunnel vision effect. First, traditional graphical systems often supply pan and zoom capabilities, where users can pan or scroll a window across a virtual canvas, and they can adjust the scale of their view (and the entire space) through zooming. The problem is that, when users are zoomed out for orientation, there is not enough detail to do any real work. When they are zoomed in sufficiently to see detail, context is lost. Second, multiple windows may be provided, each with a pan and zoom capability. Although this is reasonable for small information spaces, the many windows required by large spaces often lead to usability problems due to excessive screen clutter and window overlap. Third is the map-view strategy, where one window contains a small overview, while a second window shows a large more detailed view [Beard and Walker 1990; Smith

et al. 1989]. The overview contains a rectangle that can be moved and resized, and its contents are shown at a larger scale in the large view. Map views suffer from the extra space required for the overview and from forcing the viewer to integrate detail and context mentally.

Recent advances in computer-based information visualization have acknowledged the importance of balancing local detail with global context into a single view by providing *fisheye views* of the data space [Furnas 1986]. Analogous to a wide-angle camera lens, the idea is to show "local" detail in full (the objects of interest to the user), while displaying successively less detail for information further from the focus of attention. This can be done by three methods. First, we can graphically distort the view, where items shrink as they move away from the focus point. Second, we can present partial views through filtering, where a distance function determines whether or not items should appear on the display. Finally, we can use simpler, smaller representations and abstractions, for example, representing a detailed circuit as an icon.

Particular fisheye strategies for viewing large information spaces have been proposed and implemented by several researchers. Although most overlap somewhat in principle, they differ considerably in the type of data structure they can display, the visualization method used, and their dependency on the semantics of the application. Thus, the best choice of visualization strategy depends heavily on how the application's information is structured and on how well the visualization matches the end-user's conceptual model of the information. To get a feel for the diversity of approaches, a variety of fisheye visualization systems are summarized in Table I and discussed next.

Furnas [1986] pioneered the idea of fisheye views. He described a generalized "degree-of-interest" function, where the interest value of a node in the graph is a function of both its a priori importance and its distance from the user's current focus. He created systems for viewing and filtering structured program code, biological taxonomies, and calendars. He then verified that fisheye views were indeed superior to flat views by performing a modest usability study. Remde et al. [1987] applied the fisheye idea to SuperBook, a mostly text-based electronic book. SuperBook uses the now familiar notion of a manually expandable table of contents. Depending on how content headings are selected, subheadings are revealed or hidden. When the reader specifies a search term, SuperBook posts the number of search term hits in the free text against the headings that contain them. In essence, the hits represent the degree of interest, while the expandable contents implement the fisheye view. These ideas appear to work, as a usability study of SuperBook found that students can answer search questions with it better than with conventional text [Egan et al. 1989]. Sarkar and Brown [1992] pursued a mostly 2D graphical approach to fisheye views that distorts the position and size of nodes within a connected graph to reflect the importance of nodes. All nodes within the network are shown unless a particular node's "display" value fell below a threshold, in which case it is removed from the graph's view. Their algorithm handles

Table I. Summary of Selected Information Visualization Systems

System	Data Structure	Visualization Method
SuperBook [Egan et al. 1989; Remde et al. 1987]	Hierarchical text-based table of contents	Expandable text-based contents hierarchy; frequency of search term hits are posted next to the content headers.
Graphical fisheye views of graphs [Sarkar and Brown 1992]	Graphs	2D planar and polar transformation of graphs with filtering and multiple focal points.
Layout-independent fisheye views [Noik 1993]	Hierarchically clustered hypertext graphs	2D progressive exposure of hierarchical detail combined with fisheye space allocation and multiple focal points.
IDG hypermedia system [Feiner 1988; Feiner et al. 1982]	Hierarchically clustered hypertext graphs	2D progressive exposure of hierarchical detail combined with fisheye space allocation and multiple focal points using multiple windows.
Variable zoom (this article)	Hierarchically clustered graphs	2D progressive exposure of hierarchical detail combined with fisheye space allocation and multiple focal points.
Tree Maps [Johnson and Shneiderman 1991; Shneiderman 1991]	Strict hierarchies	2D space filling by slice and dice.
FlexView [Schaffer and Greenberg 1993]	Strict hierarchies where nodes contain several numeric attributes	Fisheye and filtered view of hierarchy; overview map window relates fisheye to global information space; dynamic queries on attributes alter the view.
Table Lens [Rao and Card 1994]	2-by-2 tables	Size and detail of a table's rows and columns matched to degree of interest; data summarized as dense bar charts.
Perspective Wall [Mackinlay et al. 1991]	Linear structures	3D perspective projection and animation.
Cone and cam trees [Robertson et al. 1991]	Strict hierarchies	3D visualization of tree and animation.

planar and polar transformations of connected graphs and uses Euclidean distance to calculate the degree of interest. Although the resulting images are impressive, Sarkar and Brown note that users sometimes perceived the resulting view as unnatural, particularly when a familiar object (such as a map) is severely distorted. The technique was later extended via a "rubber sheet" metaphor [Sarkar et al. 1993], which allows multiple foci and gives the user direct control of how much screen space is used for objects in the areas of interest. The rubber sheet approach was also adopted by Kaltenbach et al. [1991] when dealing with the problems of managing screen space in hypertext systems. Noik [1993] combined fisheye views and

hierarchical nesting of graph nodes to hypertext. Nodes (documents) are arranged and linked in a nested hierarchy (essentially a table of contents). When a bounding box of a hierarchical component is displayed, its size and detail are adjusted to reflect its degree of interest. The IDG hypermedia system [Fiener 1988; Fiener et al. 1982] visualized hierarchical clusters, allowing multiple areas of interest to be displayed at any desired level of detail in multiple windows.

Tree Maps, developed by Johnson and Shneiderman [Shneiderman 1991; Johnson and Shneiderman 1991], use a 2D-space-filling algorithm to fit a complete strict hierarchy into a window. It is based on every node containing a value that is the sum of the node values of its children. This value determines the node's relative size on the screen. One use of Tree Maps was to display a hierarchical file system, where the value shown is the size of the directories and files. Tree Maps presents a very different way of viewing information, and there are still outstanding questions on its usability. Schaffer and Greenberg [1993] developed FlexView to visualize strict hierarchies, where each node could contain a set of numeric attributes. Through FlexView's dynamic query controls, a user can select the attributes of interest, as well as the numeric ranges. The visualization shows only the subtrees that contain query hits, with hits emphasized in both color and size to represent the number of attributes matched. An overview map window provides a "gestalt" view: it contains a scaled-down version of the uncollapsed tree, indicates the extent of the hierarchy shown by the fisheye view, and displays the distribution of hits over the tree. Rao and Card [1994] developed the Table Lens for visualizing large tables. It works by adjusting the size and detail of a table's rows and columns, depending on their interest values, and by graphically summarizing its data as dense bar charts. Three-dimensional fisheye visualization was introduced by Mackinlay et al. [1991], who linearly transformed a 1D space by projecting it on a 3D "perspective wall." Robertson et al. [1991] then combined 3D effects and animation for displaying 2D hierarchies in "cone trees" and "cam trees."

Although there is much interest and intuitive appeal in fisheye views, there have been few quantitative studies evaluating its merits. Our own work applies a particular kind of fisheye view, which we call "variable zoom," to hierarchically clustered networks (explained in Section 2). These can be used to represent and view real environments—telephone systems, oil pipelines, power grids—that are controlled by operators of supervisory control and data acquisition (SCADA) systems, as well as other domains amenable to hierarchical clustering, such as hypertext [Noik 1993]. Because our interests are in critical real-time control environments, we wanted to see how well operators could navigate and manipulate a hierarchically clustered graph using either a traditional zoom method or fisheye views.

We begin with a description of the Simon Fraser variable-zoom display algorithm, which was used to provide a fisheye view interface to a simulated telephone network. We then describe a controlled experiment con-

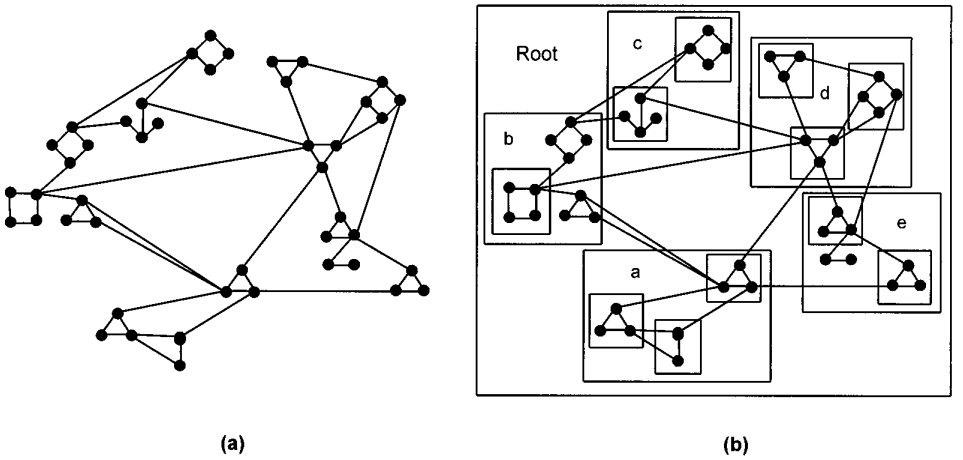


Fig. 1. (a) An example network; (b) an example of how the network can be hierarchically clustered.

trasting user performance using both a standard full-screen zoom view and a fisheye view. As well, we discuss the comments and concerns raised by highly experienced operators of very large control rooms. After presenting the results, we reexamine the experiment and the system, outline how we are now addressing the issues raised, and then suggest several implications our research has to interface design of information visualization systems.

2. THE VARIABLE-ZOOM DISPLAY METHOD FOR 2D NETWORKS

Furnas' [1986] fisheye approach was quite effective for tree structures; our work extends his ideas to 2D connected graphs as the primary data structure with superimposed *hierarchical clustering* [Fairchild et al. 1988]. Although superficially similar to Sarkar and Brown's [1992] graph visualization, it differs considerably because it visualizes hierarchical clusters in progressive detail, as well as the nodes of the graph. Our approach is most similar to Noik's [1993] parallel work on hypertext visualization, except our approach also allocates space to display the contents as well as the title of a graph node and is not tied to a particular application. It also bears similarities to Feiner's IDG hypermedia system [Fiener 1988; Fiener et al. 1982], which visualized hierarchical clusters; however, our approach provides for multiple foci within a single window. Other researchers have developed various kinds of hierarchical graph structures, such as Higraphs [Harel 1988] and Hypergraphs [Berge 1973], to which the hierarchical clusters used here bear a resemblance. However, our interests are in ways to represent graphically the structure that is of help to a user and to evaluate its effectiveness, rather than the underlying graph-structure properties of the base network.

Our variable-zoom method works on a 2D network of nodes and links, such as the one shown in Figure 1(a). It assumes that a hierarchical clustering of nodes has been superimposed on the network. Figure 1(b), for

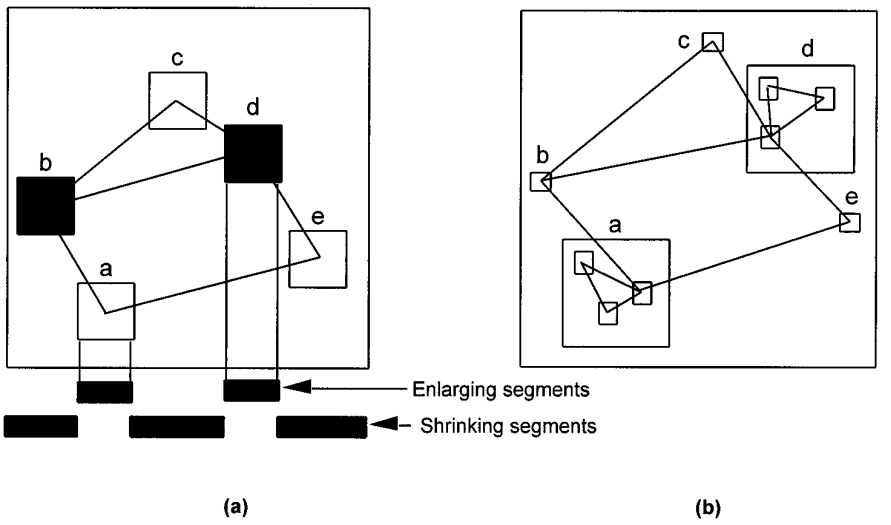


Fig. 2. Example of basic operation; (a) network before zooming; (b) nodes a and d have been zoomed to show the subnetworks; other parts are shrunk as context.

example, shows how nodes in the network of Figure 1(a) have been clustered into three hierarchical levels. The largest rectangle is the "root" cluster of the hierarchy and contains five smaller clusters a–e (the second level of the hierarchy). These, in turn, may or may not contain other clusters, nodes, or combinations thereof. The bottom of the hierarchy is reached when a cluster contains only network nodes. As long as a strict hierarchy is maintained, nodes can be clustered in any way the designer wishes, e.g., by using geographical distance, by task-specific relationships of nodes, and so on. For example, consider how this scheme could visualize the workings of an electric utility company. The "network" is the actual connection of power stations (nodes) by power lines (arcs). The superimposed clusters may represent the hierarchical geographic regions that contain the power stations, such as states or provinces, regional districts, towns and cities, and city neighborhoods, down to the power station and its subcomponents.

The hierarchy is then used by the visualization algorithm to allow clusters of the network to be viewed at different levels of hierarchical detail. At each level above the network node level, we represent the clusters as icons that may be "opened" to show the next level down. In Figure 2(a), for example, clusters a–e are drawn as icons; the links indicate that clusters are connected by at least one path in their respective subnets. Figure 2(b) shows the operation of opening (zooming into) two higher-level icons (a and d). The key to obtaining the fisheye effect is to magnify appropriate parts of lower levels uniformly so as to show detail, while embedding this detail in the remaining, uniformly scaled down network. An advantage of this method is that multiple areas of focus (detail) are allowed.

The basic variable-zoom algorithm described in Sections 2.1–2.4 assumes that all nodes (leaf and cluster) are square and do not overlap. That is, projections of nodes on x- and y-axes do not overlap. It works by applying the same method in horizontal (x) and vertical (y) directions; the description below applies to both. Nodes are in either a zoomed or unzoomed state: zoomed nodes are opened like windows, displaying their immediate subnetwork. Unzoomed nodes are closed as icons; their subnetworks are not shown. Cluster links are simply straight lines joining clusters.

In the implementation, both network nodes (leaves) and cluster nodes are represented as squares. Texturing is used to distinguish the two visually. Texturing is also used to differentiate lines representing two or more real links in the network from those representing single links. Texture is displayed using white, vertical stripes on the background color node color.

The algorithm is described for a single node, assumed to be in a zoomed state, and its subnetwork. The display size of each node or cluster in the subnetwork is calculated. Nodes and clusters to be magnified are zoomed by a magnification factor, F_e, and others are reduced in size by a shrink factor, F_s. Finally, placement of all nodes and clusters is calculated. The resulting procedure is then applied recursively to each cluster. Section 2.4 describes how to extend the algorithm to handle rectangles and overlaps, but it is not used for the system described in this article.

2.1 Magnification Factors

We first calculate F_e, the magnification factor to be applied to nodes to be zoomed, and F_s, the shrink factor to be applied to the remaining spaces. We do this by considering two ratios: R_z is the ratio of nodes to be zoomed with respect to their environment (length of parent node, L), and r is the ratio of nodes to be zoomed to the total length of all nodes, before the zoom operation is applied.

$$R_z = F_e S_z / L, \qquad 0 < R_z \leq 1, \tag{1}$$

$$r = S_z / S_a, \qquad 0 < r \leq 1,$$

where S_z = sum of lengths of all nodes to be zoomed, and S_a = sum of lengths of all nodes. The use of ratios keeps the development independent of the particular level in the overall network.

To make the subnetwork detail visible, R_z should never be smaller than some threshold value. Since a node may be arbitrarily small (and since the user may wish to zoom just one node), r can be arbitrarily small, and we need $R_z \geq threshold$ as $r \to 0$. If all nodes are zoomed ($r = 1$), no context needs to be retained, and $R_z = 1$ when $r = 1$.

As the number of nodes increases from one to the total number (in general, as r increases from zero to one), R_z should increase from the threshold to one smoothly and monotonically. A simple relationship meeting all of these requirements is $R_z = k_1 r + k_2$, where k_1 and k_2 are

constants. Because $R_z = 1$ when $r = 1$, $k_1 = 1 - k_2$; renaming k_2 as K_b (a balance factor, discussed below), we have

$$R_z = (1 - K_b)r + K_b, \tag{2}$$

and substituting into (1),

$$F_e = K_b(L/S_a)(1/K_b - 1 + 1/r). \tag{3}$$

To use space effectively, the sum of magnified and demagnified segments should equal the length of the containing node after the operation, so that

$$F_eS_z + F_s(L - S_z) = L \qquad \text{and} \qquad F_s = (1 - F_eS_z/L)/(1 - S_z/L). \tag{4}$$

The expressions for F_e and F_s indicate that both are dependent on the environment and on the user's request (i.e., on the number of nodes to examine in detail). For this reason, we refer to the algorithm as a *variable-zoom* type of fisheye method.

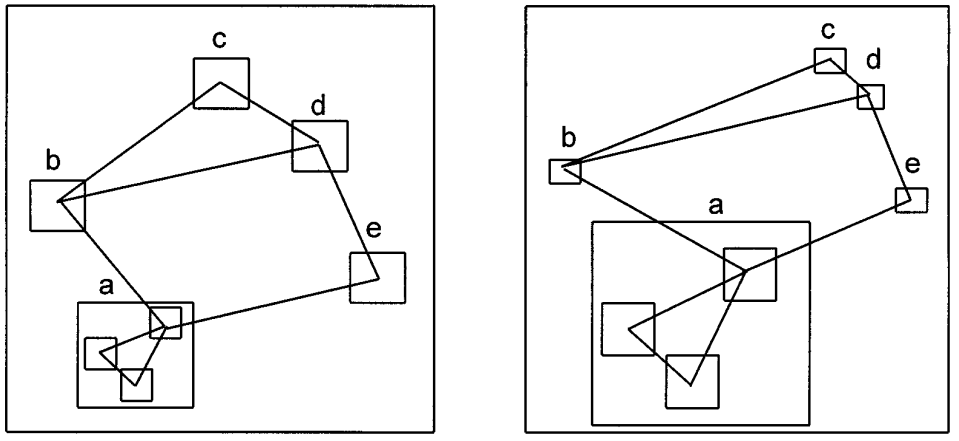
2.2 Basic Operation

The operation applied to each node recursively simply calculates the new sizes and locations. The sizes are just the original sizes multiplied by F_e or F_s, as appropriate. To calculate the positions, the x-axis is divided into segments by the boundaries of nodes to be zoomed (an identical procedure is used on the y-axis). Segments corresponding to nodes to be zoomed are enlarging segments; the others are shrinking segments (Figure 2(a)). Let x_i and x_i' be the positions before and after, l_s the length of the segment, and d_i the distance from x_i to the left boundary of the segment containing x_i. The x_i are calculated by first sorting the segment list from left to right and then performing the following for each node (the result is shown in Figure 2(b)):

initialize x_i' to the left boundary of parent node
for each segment to the left of x_i
  if enlarging, $x_i' = x_i' + F_e l_s$
  else $x_i' = x_i' + F_s l_s$
for the segment containing x_i
  if enlarging, $x_i' = x_i' + F_e d_i$
  else $x_i' = x_i' + F_s d_i$.

2.3 Balance Factor

The constant K_b in the expressions for F_e and F_s is a "balance" factor that controls the ratio of detail area to parent area, i.e., the ratio of detail to context. A larger K_b gives a larger proportion of detail. To see this, we rearrange (2) as $R_z = r + (1 - r)K_b$ so that, for a constant number of nodes to be zoomed (r is constant), R_z grows with K_b. Figure 3 illustrates the visual effects of varying K_b. When users are allowed to adjust K_b, they can control the relative emphasis on detail and context.



(a) $K_b = 0.2$ **(b)** $K_b = 0.5$

Fig. 3. The visual effects of varying the balance factor.

2.4 Extensions for Nonsquare and Overlapping Nodes

The above development assumes that all nodes are square and that their projections onto the x or y do not overlap each other. Real networks do have such overlaps, and it is often either difficult or disadvantageous from a human factors point of view to rearrange a network to avoid overlaps. Furthermore, supporting nonsquare node shapes, particularly rectangles, is desirable and sometimes necessary. This is certainly the case in most process control environments. For example, a utility company's network has bottom nodes that are essentially substation circuit diagrams. Representing these diagrams clearly while making effective use of screen space requires rectangular-shaped nodes.

Although the system described in this article uses only nonoverlapping squares, dealing with rectangles is fairly straightforward. We simply keep individual node lengths and widths, and allow for S_a and S_z being different in x and y. Handling overlap requires some additional work. The key is in the definition of S_a and S_z, as well as in allowing different values for each axis. First, a separate calculation is performed for each axis. Second, before this calculation, all nodes are projected onto the x- and y-axes, and overlapping segments are merged:

—any segment entirely within another is dropped from the calculation and
—partially overlapping segments are merged into a single segment.

Separate magnification factors are then computed for each axis (F_{ex}, F_{ey}) according to Eq. (3), and we set the final value to the minimum, $F_e = \min(F_{ex}, F_{ey})$.

Computing F_s is not quite as straightforward. We first calculate F_{sx} and F_{sy} by substituting the new value of F_e into Eq. (4). Observe that even though the "common" F_e is used, because of the dependency on S_z and

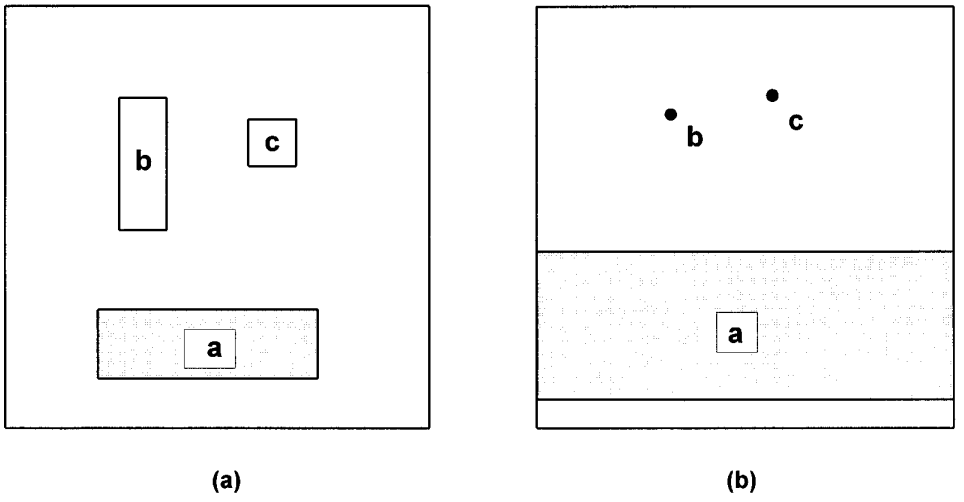


Fig. 4. Result of setting $F_s = \min(F_{sx}, F_{sy})$ when zoomed nodes include all unzoomed nodes; (a) before zooming node a. Note that since node a includes all unzoomed nodes (b and c) in their x direction $F_s = \min(F_{sx}, F_{sy}) = 0$; (b) after zooming, nodes a, b, and c reduce to points since $F_s = 0$.

because S_{zx} may be different from S_{zy}, we end up with two shrink factors, F_{sx} and F_{sy}. However, we cannot simply set $F_s = \min(F_{sx}, F_{sy})$, as we did for F_e, because either or both F_{sx} and F_{sy} may be zero. This condition occurs if, in the relevant axis, the nodes to be zoomed "include" all unzoomed nodes, as illustrated in Figure 4. The $F_s = 0$ result is easily derived by noting that the "zoomed-in nodes including all unzoomed nodes" condition is equivalent to $S_z = S_a$ and by substituting this result into (3) and (4). Neither can we set $F_s = \max(F_{sx}, F_{sy})$, since both F_{sx} and F_{sy} may be zero. To bypass these difficulties, we add the not unreasonable requirements that F_s must be greater than zero and less than one, and calculate a common F_s, as shown in Table II.

The basic operation described in Section 2.2 must also be slightly revised. Although node sizes are calculated as before (using the common values of F_e and F_s), computing the position differs in that we must use the (possibly different) values of F_{sx} and F_{sy}. The reasons for this are the axis dependence in Eq. (4) for F_s and the requirement to use screen space fully. F_e and (the common value of) F_s will not, in general, satisfy (4) in both the x and y directions. Thus, we use the algorithm of Section 2.2, but with a separate calculation for each axis, with the common F_e and axis-dependent F_s(F_{sx} and F_{sy}).

3. DESCRIPTION OF THE LABORATORY EXPERIMENT

This experiment compared the performance of subjects navigating and repairing a simulated telephone network, represented as a hierarchically clustered graph. Subjects used either a full-zoom or a fisheye view method to navigate the clusters.

Table II. Calculation of Common Shrink Factor F_s

F_{sx} and F_{sy}	F_s
both >0	min (F_{sx}, F_{sy})
one = 0	max (F_{sx}, F_{sy})
both = 0	constant between 0 and 1

A value of 0.2 for the constant yields reasonable results.

Hypothesis. Our null hypothesis was that there is no difference in performance ($p = 0.05$) between subjects traversing a hierarchically clustered network using a fisheye view or a full-zoom view. Since this is a within-subject experiment where each person uses both viewing methods, we counterbalance-ordered the methods and proposed a secondary hypothesis that order (and learning) has no effect on the outcome. A subject's performance was measured in terms of the total time taken to complete the task, number of zooms performed, and their success at performing an assigned task.

Subjects. Twenty subjects were selected from a pool of volunteers. All were senior undergraduate students, graduate students, or faculty in computer science and were familiar with graphical user interfaces and general data structures. None were familiar with the Simon Fraser fisheye view system.

In addition, two highly experienced control room operators informally evaluated the two viewing approaches under similar experimental conditions as the other subjects. Because their performance was not measured rigorously, their comments will be reported separately in the discussion section.

Materials. The experiment was performed at the University of Calgary on Sun workstations and used an implementation of the variable-zoom algorithm developed in the Computer Graphics Laboratory at Simon Fraser University. The same software could be set to display either a full-zoom or a fisheye view of a simulated telephone network. The system allowed subjects to navigate through the graphs and to change the status of links. Timing information and user events were automatically recorded.

The fisheye view used the variable-zoom method described in Section 2, with the balance factor, K_b, set to 0.5. Subjects could not adjust the balance factor. The full-zoom method followed a more traditional approach, where a selected node was enlarged to occupy the entire screen. From the users' point of view, the only differences encountered during the experiment were in the visualization method. The system interface and the hierarchically clustered graph were otherwise identical.

Both systems illustrated a hierarchically clustered simulated telephone network with four levels of substations (nodes and clusters) represented by boxes and connected by lines (Figure 5). The entire network contained 154 nodes and 39 clusters. Each node was labeled to provide a reference to the node, as well as contextual information. Figure 5(a) illustrates the root

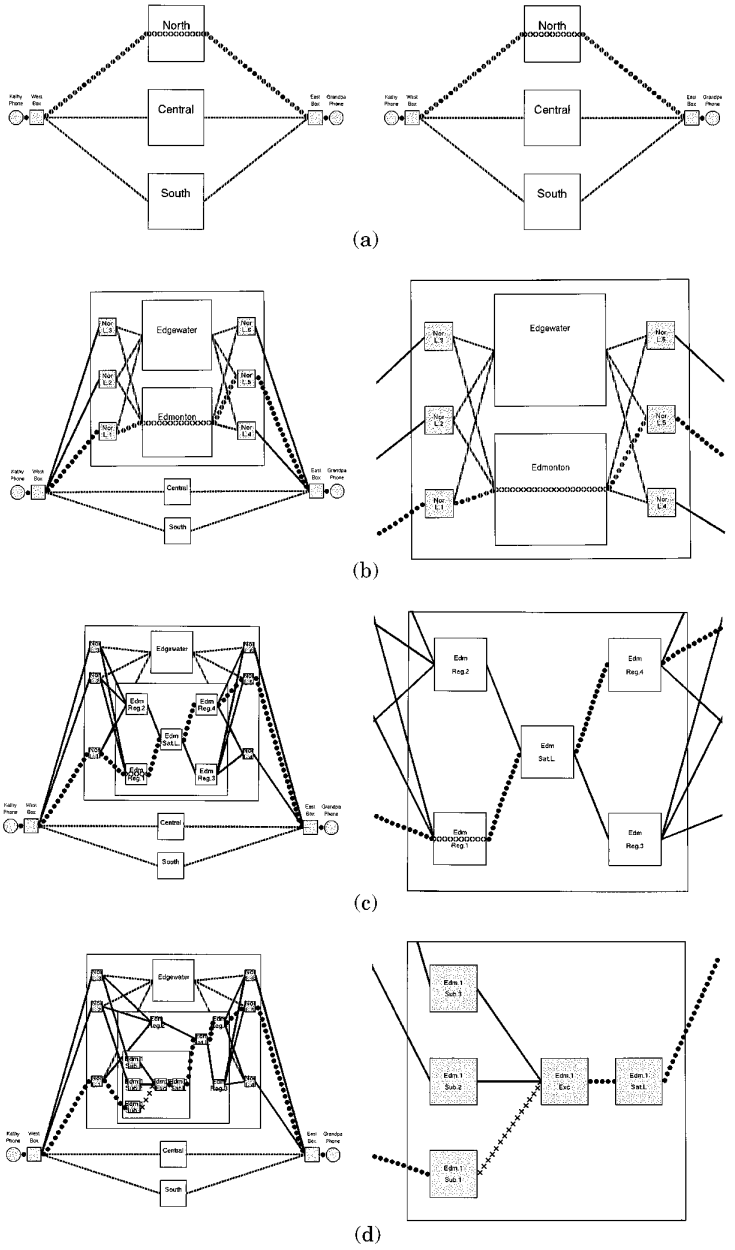


Fig. 5. Snapshots at each level in the hierarchy for the fisheye (left column) and full-zoom (right column) views. Levels are (a) root view; (b) north cluster; (c) Edmonton cluster; (d) Edmonton register 1 cluster.

view of the network, comprised of four network nodes (the phones and east and west boxes) and three expandable clusters (north, central, south). The left column shows the fisheye view and the right the full-zoom view; in the case of Figure 5(a) both views are identical, since this is the root view. When a user clicks the left mouse button on the North cluster, that cluster is exploded to the next level of hierarchical detail (Figure 5(b)). In this case, the fisheye view in the left column shows the contextual detail around the exploded cluster, while the right shows only the cluster and its entry/exit links (but it is shown larger on the display). Similarly, expanding the "Edmonton" node and then the "Edm Reg 1" node will produce the displays seen in Figure 5(c) and (d).

Color was used to provide information about telephone lines (the links), while texturing indicated user selectability of nodes and lines. Only nonleaf nodes could be selected for zooming or unzooming, and lines could only be selected if both ends were connected to leaf nodes. Selection of a line resulted in its color changing. Rerouting of telephone lines was performed by selecting or deselecting connection lines, i.e., by coloring or uncoloring the links between nodes (links between clusters were not selectable).

Task Description. Subjects were asked to act as telephone technicians. They were given a hierarchically clustered telephone network and asked to navigate through the network by zooming and unzooming nodes using the mouse.

Subjects were first asked to find a broken telephone line in the network. Breaks were displayed visually as red lines, with line texture (solid or dashed) indicating whether they connected leaf or nonleaf nodes (Figure 5). After finding the break, subjects clicked on a button labeled "Break Found." They were then asked to "repair" the network by rerouting a connection between two endpoints of the network that contained the break. Subjects then clicked on a button labeled "Reroute Done," which ended the task.

Methods. Subjects were first given a short training task to perform, where they were expected to achieve some competency with the task and the software. Subjects then performed two similar navigational tasks: "Task A," followed by "Task B." Half of the subjects performed the first task, Task A, with the full-zoom view and the second task, Task B, with the fisheye view. The remaining subjects employed the same views in the reverse order. The training task was repeated using the second view before performing Task B.

The experimental design was a 2×2 factorial design, where the two factors (the independent variables) were the view and the order in which the views were performed. Each factor had two levels: the view factor having levels "fisheye" and "full-zoom" and order having levels "fisheye-first" and "fisheye-second." The order factor is strictly an experimental artifact of within-subject design; we used it only to check for transfer effects and task differences.

The dependent variables were the time to complete the task, the number of zooms of network nodes that occurred during the task, and whether or

Table III. Means and Standard Deviations of Dependent Variables at the Statistically Significant Treatment Levels

View	Mean	Std Dev
(a) *Running Time* (*seconds*)		
Fisheye	101.9	59.5
Full-zoom	161.2	71.6
(b) *Number of Zooms*		
Fisheye	6.3	2.7
Full-zoom	10.9	4.3
(c *Successful completion of task as a ratio*		
Both	0.7	0.5

not the task was successfully completed (if subjects did not correctly repair the break, the task was considered unsuccessful). Data collection was mostly automated. The software timestamped and recorded every user event within the task (user selections, zooms, and unzooms). However, the experimenters could only determine the existence of an error by retracing each subject's actions through the hierarchy.

Qualitative comparisons between the two types of views were also gathered from subjects. We recorded their comments while they performed the experiment, and we administered a questionnaire after each task was completed. On each questionnaire subjects were asked to describe their strategy for solving the task, how they oriented themselves within the hierarchy, and what they liked and disliked about the system. After both tasks were completed, a final question asked each subject which view method they preferred using.

4. RESULTS

We analyzed the subject's running times for completing each task. An analysis of variance revealed that running time was significantly affected by the view factor used in the task ($F = 9.91$, $p = 0.01$), with people completing the task much faster when using fisheye views (102 seconds versus 161 seconds). Order did not have a statistically significant effect on the running times ($F = 3.42$, $p = 0.62$). The means and standard deviations of the running times are shown in Table III.

We also analyzed each subject's number of zooms on network nodes per task, which provides a quantitative measure of the amount of navigation required to complete the task. Note that "unzoom" actions were not analyzed, because nodes in the fisheye graph did not need to be unzoomed (since everything can remain visible on the screen). The analysis of variance revealed that the number of zooms was significantly affected by the view factor used ($F = 18.29$, $p = 0.00$), where subjects using the full zoom required almost double the number of zooms than when using the fisheye view (11 versus 6 zooms). Differences due to ordering were not significant ($F = 1.13$, $p = 0.30$). The means and standard deviations of the number of zooms are shown in Table III.

Finally, we analyzed the number of correct solutions as a percentage of the whole. This was done by simply grading each task as correct (1) or incorrect (0); other than this, we did not attempt to assign a "value" of correctness for each solution. Neither view level ($F = 2.32$, $p = 0.15$) nor order ($F = 1.12$, $p = 0.30$) had a statistically significant effect. Although most subjects did complete the task successfully, 30% of them did not (Table III, part (c)). We believe that software enhancements, such as automatic checks for completion, would have improved all correctness values.

Some other effects beyond those analyzed statistically are worth noting. First, there was little difference in the performance of subjects when locating the broken telephone line within the hierarchy using either the full-zoom or fisheye system. This is because the display, independent of the views, clearly showed which of the lowest-level clusters visible on the display contained the break (the one containing the red line). With a minimum of 4 operations required, the average of subjects using either view was 4.5 operations, with a standard deviation of 1.1.

Second, the ability of people to complete a task successfully deserves revisiting. No feedback on the condition of a path internal to a node was given—i.e., nodes only showed that paths entered into it, but did not say if the paths were connected internally. Unconnected internal paths were usually the cause of an incomplete reroute. Eight of the 20 full-zoom reroutes attempted were not successful, whereas only four of the fisheye reroutes were incorrect. Incomplete paths were lacking connections, and so the number of operations (zooms, unzooms, selections, and deselections) was artificially low. Although this suggests that running time in the full-zoom case would be artificially lowered (since there were more tasks that were not completed), we did not find much difference in practice.

Third, people who successfully completed the task produced far better reroutes through the network when using fisheye views. The raw data show that one-third of the correct reroutes using fisheye views were near the minimum possible number of operations; only a single subject had an extremely poor reroute. In contrast, no one using full zooms came close to the optimum reroute; performance was generally poorer, and a full third of the solutions were extremely poor. There was a high degree of variance among subjects in the operations performed to complete a reroute. This is discussed further in Section 5.1.

5. DISCUSSION

5.1 Examination of Results

Subjects using the fisheye view were more efficient at performing the task than when they used the full-zoom technique. In particular, they took less time to complete the task, and the amount of navigation (indicated by the number of "zoom" actions) was reduced. This corresponded well to our subjective observations during the experiment: we saw that subjects using

the fisheye view were able to focus directly on the task and were not as distracted by the need to visualize the network mentally. In the questionnaires most subjects also stated that they found the context provided by the fisheye view a valuable resource for completing the task.

The tasks consisted of two parts: finding the broken telephone line and then rerouting the connection around that line. For both systems, subjects used the same strategy—a deterministic depth-first search—for finding the broken line. However, rerouting was performed using several different strategies. Most subjects attempted to use as much of the original connection path as possible, having the reroute path be as close as possible to the original path (we call this *local rerouting*). This explains why people produced better reroutes in fisheye views, for the surrounding context easily showed them how nearby nodes were connected to each other. In contrast, subjects using full zoom had great difficulty doing local rerouting, for they became confused about their current position in the network and could not remember what nodes they had already examined. Several subjects, after attempting local rerouting, instead chose to reroute along a completely different path, starting from the top down. Although this reduced their confusion, it also required more selection of lines, which was very inefficient.

From their comments and responses to the questionnaires, most subjects greatly preferred the extra context provided by the fisheye view. It allowed them to concentrate directly on the task and reduced their cognitive burden of trying to remember the structure of the entire network. But the choice is not clear. Two of the subjects, for example, preferred the full-zoom system and said that fisheye views presented a cluttered display that was difficult to work with. Two others qualified their preference of fisheye views by saying that their choice would depend on the task complexity and the size of the network. Although these comments could arise from individual differences in visualization, all subjects did say that they had more difficulty using the full-zoom view after using the fisheye view.

5.2 Verification: What Control Room Operators Have to Say

We took the entire experimental setup to a large utility company that controlled its province-wide equipment through a central supervisory control and data acquisition system. Four to five highly experienced operators per shift run the control room, and each person is responsible for a different part of the system. Two operators agreed to participate in the experiment. However, instead of being strictly concerned with running the tasks as other subjects did, the operators were more interested in trying out the fisheye and full-zoom methods and discussing its ramifications in light of their experiences. Note that the experimental setup was not connected, or in any way related, to the normal system used by the operators.

Although we did not collect enough data to warrant a statistical study, the measured responses of the operators were in line with what we had seen in our inexperienced subjects. Both the amount of time taken and the

number of zooms required were less for the fisheye condition than for the full-zoom condition. The operators also preferred fisheye views. They said they found the variable-zoom (fisheye) technique easier to use for the task, especially because it provided an overview. They did not have difficulty keeping track of where they were in the network because, as one operator mentioned, "when you zoomed in you saw the connections." In contrast, the full-zoom technique was castigated. One operator said, "it was really confusing; I couldn't find my way. I thought it would be easy, but I had to start thinking about what was inside where." The confusion of where one was in the network, as well as the constant need to zoom in and out of layers, made it difficult for the operators to use.

However, the operators still had some concerns about fisheye views, especially because the prototype we used did not display the customary gauges and information that they expected to see (Figure 5(d)). As one person stated,

> The variable zoom was really easy to use, but the full zoom provided a lot more information. Depending on what you needed to do, the [full zoom may be] better. On grid, we could probably get by with the variable zoom. If you had to go into a sub and close a line, you'd need the [full zoom].

The "grid" refers to looking at the entire network, while the "sub" is a reference to a particular station circuit diagram (equivalent to a bottom-level node). What makes this comment interesting is that bottom-level nodes in both the fisheye and full-zoom techniques show exactly the same information, except that they are scaled differently (Figure 5(d)). The real problem is that, although this operator found the variable zoom better for navigating, he wanted to have the familiar larger-size diagrams at the bottom-level nodes, which would allow him to see clearly the essential detail in the circuit diagram. Another operator had a similar comment:

> The variable zoom lets you look into a station as well as see the big picture. However, I am not sure if it would be useful in our system because we need to get to the level of seeing analogs [actual control gauges].

In the real environment of these operators, all control situations occur within the bottom nodes. When a bottom node is reached, they require both the detail of the information presented in their schematics and the ability to manipulate the controls. In their system (and as with the full-zoom approach), these bottom nodes occupy the entire screen. Since a fisheye view sacrifices some screen space for context over detail, the operator's concern about losing space to display the essential controls is perhaps justified.

However, a few interface "tricks" can give operators the best of both worlds. Some methods for adding node detail to fisheye systems could include

—a second display for showing a selected node in full;
—a toggle for going between the fisheye view and full zoom on a single display;
—a dynamic balance factor, so that operators can adjust (perhaps with a scroll bar) the balance between overview and detail; and
—pop-up controls attached to gauges for doing detailed work.

The next section describes the specific ways we have addressed the operators' concerns in a new prototype.

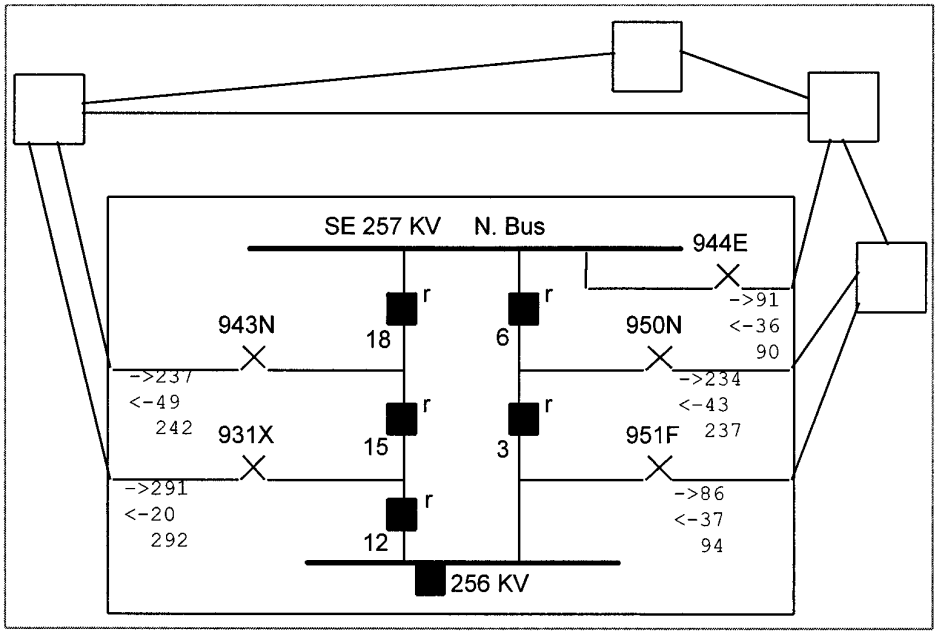
5.3 Toward a Real System for Control Rooms: Work in Progress

The bottom-level nodes in the operators' utility network are substation circuit diagrams, and as noted in their comments, these must be large enough to be understood. Furthermore, the operators use these displays to issue control commands (e.g., open or close a circuit breaker) and to obtain detailed information, such as the status of breakers, switches, and transformers, and voltage levels.

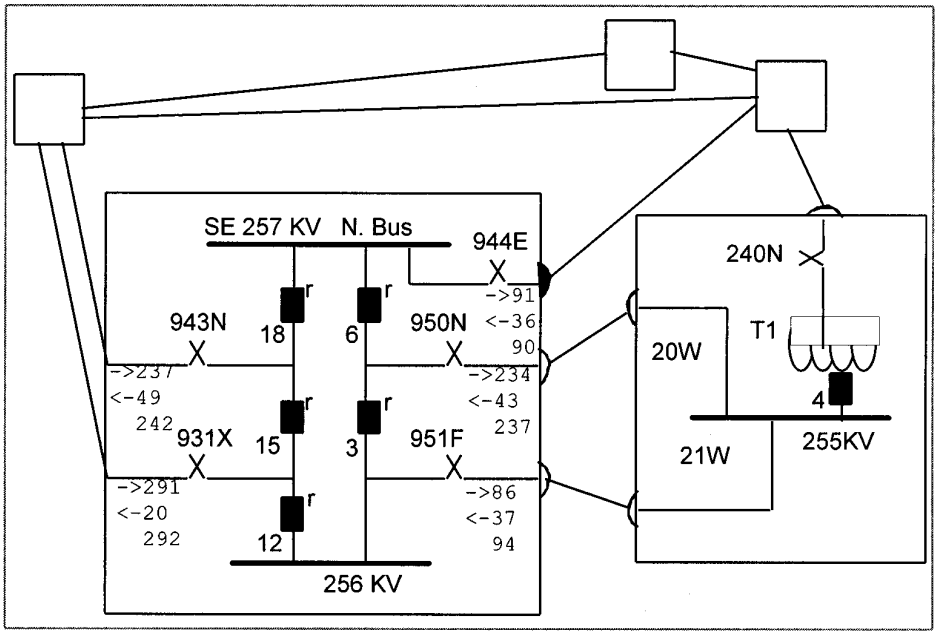
In the new prototype being developed for a utility company, we support this requirement for a large bottom-level node. To each node representing a single substation, we added a single child node containing the circuit diagram for that substation—an active display that shows all required status values and supports operator interaction to enact control commands. Zooming into the next-to-bottom node produces a readable substation schematic (Figure 6). With our current display size and layout, up to two or three substation circuit diagrams can be shown, in context, in readable form. Although the algorithm does not limit the number of circuit-level nodes, attempting to show more than a few results in displays too small to show discernible detail easily. However, since many real problem situations encountered by operators involve three or fewer stations, this does not seem a severe limitation.

It is not uncommon for real problems to arise involving two directly connected substations. With their current system, operators must flip back and forth between displays, trying to form a mental image of the problem, as well as of the context. To help with this, we have added a feature where selecting a line connecting two substations will zoom both substations to the circuit diagram level. Thus, operators immediately see both substations at the desired level, as well as the surrounding context. We believed that this would significantly improve their ability to diagnose and correct network problems. Operators have tried this and have commented favorably on this feature.

Two additional features were added to our algorithm to adapt it to the power utility's use. First, we added the ability to deal with multiple lines between a single pair of substations, since this occurs in several cases. To avoid lines falling on top of one another, we specify connection points where a line attaches to a node. In the current implementation, all lines are single straight line segments from node to node. This results occasionally in



(a)



(b)

Fig. 6. Node expanded to circuit diagram level; (a) one expanded node; (b) two expanded nodes, with scaling.

portions of lines being covered by one or more nodes, which can cause confusion (i.e., does the line enter the node or pass underneath it?). To alleviate this, a small disk shows connection points. While this works, there are other approaches. For example, conventional circuit diagrams solve this problem nicely by using multisegment lines, usually with "Manhattan" geometry (all segments are horizontal or vertical). However, maintaining Manhattan geometry links while dynamically changing node size and location is an extremely difficult and computationally demanding problem. We are working on ways of maintaining sufficiently simple link paths to meet operators' needs, while avoiding the time-consuming search space of a complete solution.

We believe that the initial concerns of the operators, whose prior experiences were with equivalencies of full-zoom systems, arose from them being unaware of the many ways detail can be brought out in fisheye views. We later showed operators the newer prototype fisheye view system that had bottom-level nodes that displayed the kind of controls they expected to see, as described above. They said that this was exactly what they wanted.

5.4 Limitations of the Experimental System

Several problems and limitations became evident during the design and execution of this experiment. These included software maturity, design of a suitable telephone network, naming problems (i.e., creating a short but mnemonically useful name for a cluster), and problems with deciding how to represent "composite" edges in the graphs. Although none are serious enough to compromise the experiment, they do indicate areas for improvement.

The software we used in the experiment was still under development; Simon Fraser University had created a special version to accommodate scheduling constraints at the University of Calgary. Thus, the software had several limitations that we expect to be repaired in future versions. One minor distraction was the unnecessary accuracy required to operate the system: mouse clicks had to be exact, and accidental double-clicks occasionally resulted in errors. A more serious distraction to subjects concerned how screens were redrawn after an interaction request. When subjects clicked on any edge to change its color, or on any node to expand or contract it, the entire screen was redrawn with the new image. Some users commented on the abruptness and sense of discontinuity in the response to a zoom request; the simultaneous jump in position and change in size of many nodes can be disconcerting. A better approach would be to smooth the visual changes to the displayed network through continuous animation, as done in cone trees or the perspective wall [Mackinlay et al. 1991; Robertson et al. 1991]. Although the usefulness of animation in these systems has not been formally evaluated, it has intuitive appeal, and we believe this may alleviate a user's disorientation when zooming and unzooming nodes. We are now incorporating animation into a new version of variable zoom called "continuous zoom" [Bartram et al. 1994].

A secondary problem is how to label nodes and cluster icons in the network. We first tried a scheme that provided a single-letter descriptor for every node/icon that indicated its position in the hierarchy. For example, a node at level four had a four-character name plus an extension giving the type of node. This proved too cryptic, and we then tried to use longer and more meaningful labels. The tradeoff is that, while the longer labels were easier to recognize and remember, they took up too much space on the display. We finally used a compromise scheme, where longer names were used, but with the number of characters displayed proportional to the screen area occupied by the node. This approach has also been used by Feiner [1988]. We do not know how good this solution really is. Although not critical to this experiment, we expect node labeling to be important in some application domains.

Another design issue was the distinction between selectability and "compositeness." Texturing indicated that a line or node was not selectable (Figure 5). However, this meant that both leaf nodes and "composite" lines (those with one or both ends at nonleaf nodes) were textured. Some confusion resulted when subjects correlated texturing not with selectability, but with whether the object was composite or not.

All of these limitations were present in both the fisheye and full-zoom tasks. We would not expect their disappearance in future systems to change the results described in this article.

5.5 Impact for Practitioners

Our results have implications for interface designers of large information systems that are structured as connected graphs. We suggest that the designer should consider if the information can be naturally represented to the user as a clustered hierarchy. If so, we believe that users are better able to manage the information space when the display provides both local detail and global context, as is done through fisheye views.

There are surprisingly many real-world situations meeting these criteria. The particular telephone network and task used in our experiment are just one instance of the tasks generally found in many control rooms. Operators of real-time supervisory control and data acquisition systems often deal with hierarchically clustered networks such as power grids (mentioned in this article), machine plants, telephone systems, and gas pipelines. Operators must monitor the network operation. When something goes wrong with the network operation, alarms are sounded. Operators must then quickly isolate and repair problems; these are sometimes due to isolated failures of network components or could result from an interrelated breakdown of many components. Failure of these systems can affect large numbers of people, use expensive resources, and even be life critical (e.g., a nuclear power plant operation). The operator must be able to navigate through these structures quickly and accurately.

Of course, the hierarchical clustering superimposed on the network should present a good conceptual model to the user/operator. If nodes

represent, say, machinery scattered across a country, then geography could describe the hierarchical clusters—country, provinces, regions, districts, and so on. Whatever the representation chosen, it should be understandable to the people who use it, preferably in their own language and constructs.

Other situations where local detail and global context are important can be found in our daily computer usage. Much of the information we store in computers is hierarchical, such as computer file systems. Many graphical user interfaces to our file systems permit users to view several directories at once. However, these are often an all-or-nothing affair. Views do not show relations between directories (e.g., when links are allowed), and all information is shown at full size. Fisheye views could show these relations and could give more visual emphasis to user's current items of interest (e.g., Egan et al. [1989] and Schaffer and Greenberg [1993]).

5.6 Critical Reflection

Our experimental results have shown that fisheye views provide a significant advantage over conventional full-zoom views. Although the results are dramatic, some questions still remain that could be addressed in further studies; these would give us more certainty on how well we could generalize our results. In this section we consider whether our results have been compromised by the choice of subjects, choice of tasks, training time, and the "degree" of the fisheye view.

The subjects were drawn from a pool of senior undergraduate and graduate students, as well as faculty in an academic computer science environment. This group had a great degree of familiarity with both graphical interfaces and complex data structures (but not to control rooms). These traits do not necessarily generalize to other populations. Whereas the mapping of the task to an abstract network representation presented little difficulty to our subjects, other groups may find the mapping somewhat unnatural or confusing. It is possible that using a fisheye view—providing *more* information—may increase this confusion, especially if the hierarchical model is unfamiliar to the user. This could be important when fisheye views are incorporated into rarely used programs and when the hierarchy chosen is obtuse. However, we believe this problem could be mitigated and even eliminated if the hierarchical model matches the user's conceptual model of the system. As well, training time for users of frequently used systems (such as control room operators) would likely familiarize them with the general ideas. We have already seen that the two experienced operators did not have any problem navigating the telephone system network, even though the task and domain were unfamiliar to them.

The two tasks used in the experiment, both involving maintenance of a telephone network represented as a hierarchical graph, arguably generalize to other situations and domains. We did ensure that the tasks involved navigation both from the root of the graph and from deep within the graph.

We believe this to be a typical problem and suggest that a large class of tasks could benefit from fisheye techniques. However (as noted by the operators), we did not include a problem on managing controls within a bottom node, a typical control room task. Still, we have shown that this class of problems could be handled by the fisheye view technique at least as well as in a full-zoom system, as long as suitable interface strategies are incorporated to show detail at bottom levels (Sections 5.2 and 5.3). Although operator reaction is positive, further testing is required.

Training time in our experiment was minimal. Although we do expect situations where users must be able to use a system with minimal training, there are also cases where significant training is the norm (as with the control room operators). There is, of course, a possibility that the user performance differences between fisheye and full-zoom views may be reduced (or increased!) after significant training time.

Not all fisheye systems will give the same view. In this experiment we used the variable-zoom method, a particular style of fisheye views. However, there are many different fisheye views possible, even within the same system, by varying the *degree-of-interest* [Furnas 1986] function to emphasize or deemphasize the full-zoom or fisheye quality of the display. Here, this was determined ahead of time by the experimenters on an arbitrary basis. Fisheye views present a tradeoff—global context versus local detail. Also, the balance factor mentioned in Section 2 may be varied as well. While we believe fisheye views are superior to full zooming, it is unclear exactly *which* fisheye view is appropriate for a given task. By incorporating different degree-of-interest functions and balance factors as further variables in future experiments, an optimal tradeoff may be determined.

5.7 Research Agenda

The data gathered here provide encouraging results toward the use of fisheye views for navigational tasks. Research, however, is far from complete. For example, how large can the network be before information overload becomes a problem, even using fisheye views? Will increased clutter undermine the benefits fisheye views provide? Is there an optimum number of hierarchical levels for a given network size and structure?

A few subjects expressed a preference for the simplicity of full-zoom views, because the amount of information presented on-screen was always small. At the other extreme, the entire fisheye view network could be viewed on-screen simultaneously (i.e., all clusters are expanded). The 154 nodes and 39 clusters in the simulated telephone network are near the limit of what could effectively be presented at once! Consider a cluster that has been expanded. Of the nodes and icons now visible, only a few may be truly useful to the task at hand. Perhaps information filtering, an extension of Furnas' [1986] degree-of-interest function, might make fisheye views more effective by pruning the "less-useful" information from the display. Indeed, filtering is used heavily by many of the fisheye systems summarized in Table I.

We have already mentioned that the fisheye view may require alternate interface strategies to show detail at bottom levels. An interesting experiment would contrast techniques such as dynamic balance factors, pop-up windows, pop-up controls, alternate screens, and so on.

Multiple foci (independently zoomed nodes) were allowed by the software in this experiment, though their use was not examined. The potential benefits and/or problems of multiple foci are unclear and need to be examined.

In terms of the tasks being tested, improvements to the system might include an unobtrusive "spontaneous-interest" indicator. The motivation for this comes from human visual perception, where motion, even far from the focal point, is a key determiner of interest. A gentle but persistent motion (e.g., vibration) might be used to indicate a problem that may otherwise be hidden in a reduced path or node. Motion would tend to stand out well in an otherwise static display and would highlight trouble spots needing attention. Motion might better attract an operator's attention than, say, a color change.

Finally, network diagrams currently used by the operators (as well as most circuit and wiring diagrams) follow a Manhattan geometry, as mentioned in Section 5.3. We feel it may therefore be desirable to follow this convention for showing leaf node schematics in the variable-zoom algorithm, and we are currently attempting to develop such a method.

6. CONCLUSIONS

We described the variable-zoom algorithm for generating fisheye views of hierarchically clustered networks. We then outlined our experiment, contrasting fisheye views with traditional full-zoom views. Results suggest that the greater context provided by fisheye views significantly improves a user's performance of the tasks. Using the fisheye view, subjects are able to concentrate directly on the task itself, resulting in quicker navigation and less unnecessary exploration. Although real control room operators were concerned about seeing and manipulating detailed schematics in the smaller leaf nodes displayed by fisheye views, we have described several ways this can be ameliorated. We suggest that fisheye viewing interfaces should be favored over traditional viewing approaches for displaying large information spaces. Further work remains to determine how the significant advantages of fisheye views reported here will generalize across different levels of task complexity and to other data structures.

ACKNOWLEDGMENTS

The graduate course of Human Computer Interaction at the University of Calgary provided feedback to the experiment. Our subjects freely gave their time and energies; they deserve a special thanks. Troy Brooks of Simon Fraser's Computer Graphics Laboratory developed much of the data-logging software.

REFERENCES

BARTRAM, L., OVANS, R., DILL, J., AND HAVENS, W. 1994. Intelligent graphical user interfaces for time-critical systems. In *Proceedings of Graphics Interface* (Banff, Canada, May). Morgan Kaufmann, San Mateo, Calif.

BEARD, D. V. AND WALKER, J. Q. 1990. Navigational techniques to improve the display of large two-dimensional spaces. *Behav. Inf. Tech. 9,* 6, 451–466.

BERGE, C. 1973. *Graphs and Hypergraphs.* North-Holland, Amsterdam.

EGAN, D. E., REMDE, J. R., GOMEZ, L. M., LANDAUER, T. K., EBERHARDT, J., AND LOCHBAUM, C. C. 1989. Formative design-evaluation of SuperBook. *ACM Trans. Inf. Syst. 7,* 1, 30–57.

FAIRCHILD, K. M., POLTROCK, S. E., AND FURNAS, G. W. 1988. SemNet: Three-dimensional graphic representations of large knowledge bases. In *Cognitive Science and Its Application for Human–Computer Interface,* R. Guindon, Ed. Elsevier, New York, 201–233.

FEINER, S. 1988. Seeing the forest for the trees: Hierarchical display of hypertext structure. In *Proceedings of the Conference on Office Information Systems 1988,* (Palo Alto, Calif., Mar. 23–25). ACM, New York, 205–212.

FEINER, S., NAGY, S., AND VAN DAM, A. 1982. An experimental system for creating and presenting interactive graphical documents. *ACM Trans. Graph. 1,* 1, 59–77.

FURNAS, G. W. 1986. Generalized fisheye views. In *Proceedings of ACM CHI 86 Conference on Human Factors in Computing Systems* (Boston, Mass., Apr. 13–17). ACM Press, New York, 16–23.

HAREL, D. 1988. On visual formalisms. *Commun. ACM 31,* 5 (May), 514–530.

JOHNSON, B. AND SHNEIDERMAN, B. 1991. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Proceedings of IEEE Visualization 91* (San Diego, Calif., Oct. 22–25). IEEE, New York, 284–291.

KALTENBACH, M., ROBILLARD, F., AND FRASSON, C. 1991. Screen management in hypertext systems with rubber sheet layouts. In *Proceedings of Hypertext 91* (San Antonio, Texas, Dec. 15–18). ACM, New York, 91–105.

MACKINLAY, J. D., ROBERTSON, G. G., AND CARD, S. K. 1991. The perspective wall: Detail and context smoothly integrated. In *Proceedings of ACM CHI 91 Conference on Human Factors in Computing Systems* (New Orleans, La., Apr. 28–May 2). ACM Press, New York, 173–179.

NOIK, E. G. 1993. Exploring large hyperdocuments: Fisheye views of nested networks. In *Proceedings of the ACM Conference on Hypertext and Hypermedia* (Seattle, Wash., Nov. 14–18). ACM Press, New York, 192–195.

RAO, R. AND CARD, S. K. 1994. The Table Lens: Merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *Proceedings of the ACM CHI 94 Conference on Human Factors in Computing Systems* (Boston, Mass., Apr. 24–28). ACM Press, New York, 318–322.

REMDE, J. R., GOMEZ, L. M., AND LANDAUER, T. K. 1987. SuperBook: An automatic tool for information exploration—Hypertext? In *Proceedings of the ACM Hypertext 87 Conference* (Chapel Hill, N.C., Nov. 13–15). ACM Press, New York, 175–188.

ROBERTSON, G. G., MACKINLAY, J. D., AND CARD, S. K. 1991. Cone trees: Animated 3D visualizations of hierarchical information. In *Proceedings of ACM CHI 91 Conference on Human Factors in Computing Systems* (New Orleans, La., Apr. 28–May 2). ACM Press, New York, 189–194.

SARKAR, M. AND BROWN, M. H. 1992. Graphical fisheye views of graphs. In *Proceedings of ACM CHI 92 Conference on Human Factors in Computing Systems* (Monterey, Calif., May 3–7). ACM Press, New York, 83–91.

SARKAR, M., SNIBBE, S., TVERSKY, O., AND REISS, S. 1993. Stretching the rubber sheet: A metaphor for visualizing large layouts on small screens. In *Proceedings of UIST 93* (Atlanta, Ga., Nov. 3–5). ACM, New York, 81–91.

SCHAFFER, D. AND GREENBERG, S. 1993. Sifting through hierarchical information. In *Proceedings of Posters and Short Papers: ACM INTERCHI 93 Conference on Human Factors in Computing Systems* (Amsterdam, Holland). ACM Press, New York.

SCHAFFER, D., ZUO, Z., BARTRAM, L., DILL, J., DUBS, S., GREENBERG, S., AND ROSEMAN, M. 1993. Comparing fisheye and full-zoom techniques for navigation of hierarchically clustered networks. In *Proceedings of Graphics Interface 93* (Toronto, Canada, May 19–21). Morgan Kaufmann, San Mateo, Calif., 87–96.

SHNEIDERMAN, B. 1991. Tree-maps. Excerpts from the *IEEE 1991 Visualization Video*. IEEE, New York.

SMITH, R. B., O'SHEA, T., O'MALLEY, C., SCANLON, E., AND TAYLOR, J. 1989. Preliminary experiences with a distributed, multi-media, problem environment. In *Proceedings of the 1st European Conference on Computer Supported Cooperative Work* (*EC-CSCW 89*) (Gatwick, U.K., Sept. 13–15). 19–34.

Received February 1993; revised May 1994; accepted January 1995