

**VARIABLE ZOOM:
A NEW DISPLAY TECHNIQUE FOR
LARGE HIERARCHICAL NETWORKS**

by

Zhengping Zuo

M.Sc., Academia Sinica, Beijing, China, 1982

B.Sc., Tsinghua University, Beijing, China, 1968

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE

in the School
of
Engineering Science

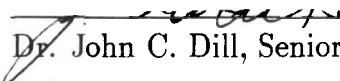
© **Zhengping Zuo** 1992
SIMON FRASER UNIVERSITY
December 1992

*All rights reserved. This work may not be
reproduced in whole or in part, by photocopy
or other means, without the permission of the author.*

APPROVAL

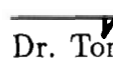
Name: Zhengping Zuo
Degree: Master of Applied Science
Title of thesis: VARIABLE ZOOM: A NEW DISPLAY TECHNIQUE FOR LARGE HIERARCHICAL NETWORKS

Examining Committee: Dr. Shahram Payandeh, Chairman



Dr. John C. Dill, Senior Supervisor

Dr. John Jones, Supervisor



Dr. Tom Calvert, External Examiner

Date Approved: 15 Dec 92

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/~~Project/Extended Essay~~

Variable Zoom: A New Display Technique for Large Hierarchical Networks

Author: _____
(signature)

Zhengping ZUO
(name)

Dec. 15, 1992
(date)

ABSTRACT

Large hierarchical networks are widely used in supervisory control systems, such as those in telecommunications network management, power distribution, and industrial process control. These networks are becoming increasingly difficult for operators to understand and operate correctly.

To help overcome navigation problems encountered in traditional methods for displaying such networks, this thesis proposes a new display technique, called a **variable zoom**, to represent large hierarchical networks. It combines hierarchical representations with a generalized fisheye strategy pioneered by G.W. Furnas. The variable zoom “nests” one or more detailed views into one or more levels of context in order to improve navigation and user understanding. The detailed regions are uniformly magnified while the levels of context are gradually demagnified. The magnification/demagnification factors are adjusted according to the environment and the user’s requests. Two prototypes of the variable zoom method have been completed, one for an abstract network, the other for a human factors experiment.

A preliminary human factors experiment showed that the variable zoom technique was significantly better than a traditional uniform zoom technique in this experiment.

To

My Family

ACKNOWLEDGMENTS

My sincere thanks to my supervisor Dr. John C. Dill for providing me the opportunity to work on this challenging and interesting project, for his many helpful suggestions, and incredible patience in reading many revisions of this thesis.

I would like to thank Lyn Bartram and Troy Brooks for their kind help, and also Shelli Dubs, Mark Roseman, and Doug Schaffer for allowing me to include their human factors experiment results in this thesis. Also, many thanks to the graduate secretary, Brigitte Rabold, for her kind help.

I am very grateful to my committee members, Dr. Tom Calvert, Dr. John Jones, and Dr. Shahram Payandeh, for their assistance and help.

CONTENTS

ABSTRACT	iii
ACKNOWLEDGMENTS	v
LIST OF FIGURES	x
LIST OF TABLES	xi
VARIABLES	xii
1 INTRODUCTION	1
1.1 Large Hierarchical Networks	1
1.2 Problem Statement	3
1.3 Graphic User Interfaces	4
1.4 Thesis Goal	6
2 LITERATURE REVIEW	9
2.1 Traditional Method and Navigation Problem	11
2.2 Map View Method	12
2.3 Fisheye View (Distorted view)	14
2.3.1 DOI (Degree of Interest) Approach	14
2.3.2 Fisheye Lens Transformation	18
2.3.3 3D Perspective Projection	20

2.4	Hierarchical Abstraction and Representation	24
2.5	Variable Zoom: A New Technique	25
3	VARIABLE ZOOM TECHNIQUE	27
3.1	Fundamental Ideas	28
3.1.1	Hierarchical Organization and Representation	28
3.1.2	Fisheye View Strategy	31
3.2	Conventions and Definitions	32
3.2.1	Conventions	32
3.2.2	Basic-operation	34
3.3	Parameters in the Basic-operation	37
3.3.1	S_a and S_z	37
3.3.1.1	Calculating S_a and S_z	38
3.3.1.2	Examples	40
3.3.2	Ratio Parameters	40
3.3.3	Enlarging Factor F_e	43
3.3.3.1	Derivation	43
3.3.3.2	Calculation	45
3.3.4	Shrinking Factor F_s	46
3.3.5	Balance Factor K_b	48
3.4	Algorithm for Basic-Operation	51
3.4.1	Transforming Node's Size	51
3.4.2	Transforming Node's Position	52
3.4.3	An Example	54

3.5	Varying Balance Factor with Level	59
4	A PRELIMINARY EXPERIMENT	62
4.1	Description of the Experiment	63
4.1.1	Zoom Methods	63
4.1.2	Task	66
4.1.3	Subjects and Experiment Method	68
4.2	Results	69
5	DISCUSSION	72
5.1	Comparisons with Other Fisheye View Methods	72
5.2	Future Work	77
6	SUMMARY	80
A	ALGORITHM FOR $F_{s,xy}$	83
B	DERIVATION OF \mathbf{K}_b FORMULA	87
B.1	Definitions and Assumptions	87
B.2	Derivation	89
B.3	Result	91

LIST OF FIGURES

1.1	A very simple hierarchical telephone network [Meunier 88].	3
2.1	The taxonomy of methods for displaying large 2D information spaces.	10
2.2	Using the map view method to display a large binary tree [Beard 90]	13
2.3	DOI function for a tree structure information space [Furnas 86]. . . .	16
2.4	An example of fisheye lens transformation [Sarkar 92].	19
2.5	Sketch of a Perspective Wall [Mackinlay 91].	22
2.6	Sketch of a Cone Tree [Robertson 91b].	23
3.1	A hierarchical organization and representation for a network.	29
3.2	Tree representation of hierarchical nodes.	30
3.3	One-dimensional view of the variable zoom layout.	32
3.4	Examples showing the basic-operation.	35
3.4	(continued).	36
3.5	An example calculating S_a and S_z	41
3.6	$R_z = K_b + (1 - K_b) \cdot (r_4)^u$	45
3.7	$R_z = K_b + (1 - K_b) \cdot r_4$ with $K_b = 0.2, 0.5, 0.8$ respectively.	49
3.8	Comparison of layouts with different K_b	50

3.9 (a)	A normal layout before a basic-operation.	55
3.9 (b)	A variable zoom layout after the basic-operation.	56
3.10	Compare constant K_b with variable K_b	60
4.1	Examples of the FZ views in the human factors experiment.	64
4.2	Examples of the VZ views in the human factors experiment.	65
4.3	An example of bad link indicator.	67
4.4	Human factors experiment results [Schaffer 92][IGI 92-0061].	70
5.1	The variable zoom method and DOI values.	75
5.1	(continued).	76
A.1	Calculating $F_{s,xy}$ when $F_{s,x} = 0$ and/or $F_{s,y} = 0$	84

LIST OF TABLES

3.1	Calculating parameters used in the variable zoom algorithm.	42
3.2	Transformation of nodes' sizes.	57
3.3	Determining enlarging and shrinking segments and building the sorted all-segment list.	57
3.4	Transformation of a node's position.	58
3.5	The $K_{b,i}$ values used in the example of varying K_b method.	61
4.1	The arrangement of the human factors experiment.	68

VARIABLES

F_e	Enlarging factor; $F_e > 1$.
$F_{e,x}$	Enlarging factor in the x direction.
$F_{e,y}$	Enlarging factor in the y direction.
$F_{e,xy}$	Common enlarging factor; $F_{e,xy} = \text{minimum}(F_{e,x}, F_{e,y})$.
F_s	Shrinking factor; $1 > F_s \geq 0$.
$F_{s,x}$	Shrinking factor in the x direction.
$F_{s,y}$	Shrinking factor in the y direction.
$F_{s,xy}$	Common shrinking factor; $1 > F_{s,xy} > 0$; in most case, $F_{s,xy} = \text{minimum}(F_{s,x}, F_{s,y})$.
K_b	Balance factor; $1 > K_b > 0$.
$K_{b,i}$	Balance factor at the i^{th} level.
L_p	length of the Parent-node.
r_1	$\equiv L_p/S_a$; $(r_1 \geq 1)$
r_2	$\equiv S_a/S_z$; $(r_2 \geq 1)$
r_3	$\equiv S_z/L_p = 1/(r_1 \cdot r_2)$; $(0 < r_3 \leq 1)$
r_4	$\equiv S_z/S_a = 1/r_2$ $(0 < r_4 \leq 1)$
R_z	Ratio between the sum of enlarging segments and their immediate Parent-node, in the variable zoom view.
	Sum of segments which are possible to be zoomed and without overlap between each other.
$S_{a,x}$	S_a in the x direction.
$S_{a,y}$	S_a in the y direction.

S_z Sum of segments which are **actually** to be zoomed-in and without overlap between each other.

$S_{z,x}$ S_z in the x direction.

$S_{z,y}$ S_z in the y direction.

CHAPTER 1

INTRODUCTION

1.1 Large Hierarchical Networks

We are surrounded by and dependent upon networks: from physical networks, such as telecommunication networks, computer networks, highway networks etc., to invisible networks, such as economic networks describing the flow of goods and services, and immigration networks representing the flow of people as they move from place to place.

Often people organize **large**¹ networks into hierarchical structures, because:

¹This thesis takes the user interface perspective, where “large” (e.g. “large networks”, “large information spaces”) refers to networks and information spaces far too large to fit onto a single computer display screen.

1. Hierarchies are easily perceived and appear in many different applications. Examples include structured programming languages, hierarchical file systems, and structured science and technological knowledge domains such as biological taxonomies.
2. Organizing information hierarchically often improves the quantity of information processed [Resnikoff 89].
3. In many cases, arbitrary networks can be transformed into hierarchies [Robertson 91b].

Figure 1.1 illustrates a very simple hierarchical representation of the United States telephone network. Usually a large network includes thousands of nodes and links. For example, BC Tel's (B.C. Telephone Company) regional telephone network comprises over 500,000 individual telephones and 1,000 trunk groups (a trunk group is a collection of many individual telephone circuits) [IGI 90].

Though large networks have become increasingly important, human-computer interfaces for such networks have not kept up, and many problems urgently require solutions. In this thesis, we focus on representation methods for large hierarchical networks used in supervisory control systems, such as those in telecommunication network management, power distribution, and industrial process control (e.g. pulp and paper mill wide control).

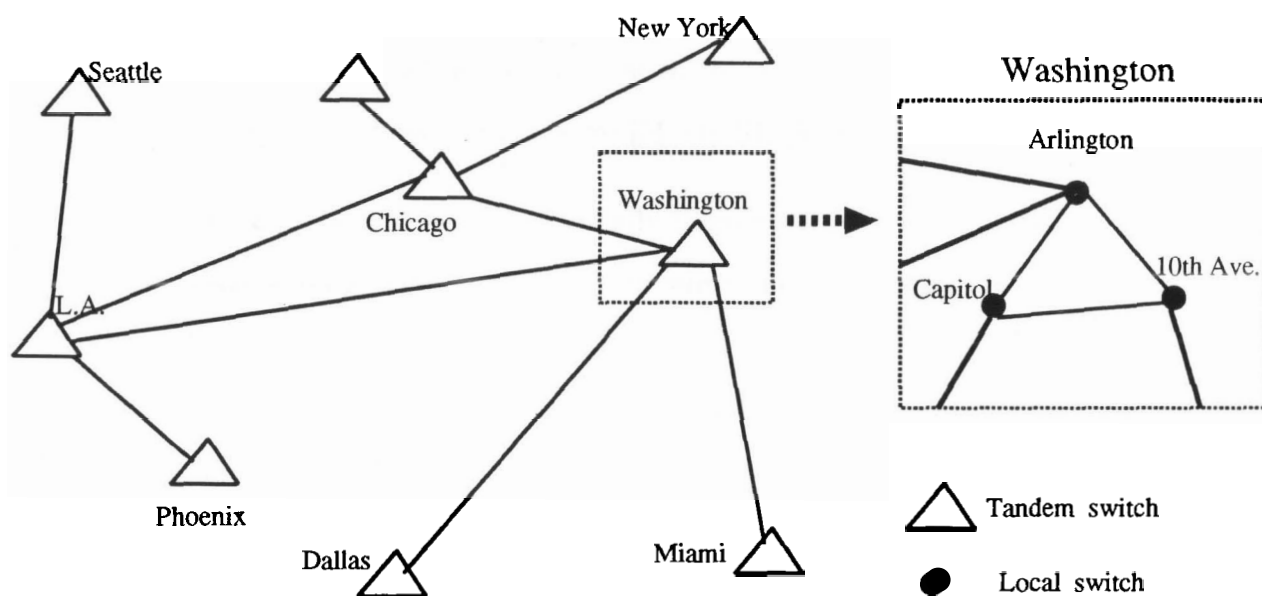


Figure 1.1: A very simple hierarchical telephone network [Meunier 88].

1.2 Problem Statement

Large supervisory control systems are becoming increasingly difficult for operators to understand and operate correctly. An “**information bottleneck**” exists between operators and systems [Havens 91]. Due to the importance of the operator’s role in these systems, the quality of interfaces between human and computer is crucial.

The primary job of the operator of a supervisory control system is identifying, analyzing and solving problems in a short time when they happen. Existing user interfaces often present relative information in a form that requires the operator to select, process, and integrate low-level information from multiple sources quickly and accurately. This behaviour demands mental capacities from the operator that violate a widely accepted limitation of human cognition. Under ideal conditions, there is a

working memory capacity limitation of no more than five to nine “chunks” [Miller 56], and the operator engaged in tasks similar to those found in dynamic control systems has a memory span of even fewer items [Moray 81] [Wickens 84].

It is difficult to represent a physically large system effectively with a relatively small display surface. Context can be lost when detailed images are being examined. This is known as the **keyhole** phenomenon: the operator can “get lost” in multiple display screens, assuming the typical scenario where each display screen permits only a small window into the larger information space [Hollnagel 83]. It is difficult to solve this problem by simply selecting bigger and better display screens. Rather, the display system should be designed to facilitate the user’s ability to extract information effectively across displays [Mitchell 87] [IGI 90].

1.3 Graphic User Interfaces

Significant and widespread interest in the quality of user-computer interfaces is relatively new in the formal study of computers.² The emphasis prior to the early 1980s was on optimizing two scarce hardware resources, computer time and memory. Because of plummeting hardware costs and increasingly powerful graphics-oriented computing environments, we can today afford to optimize user efficiency rather than computer efficiency [Foley 90].

²However, it should be observed that early work on interactive graphic user interfaces began almost as early as computer graphics itself [Newman 68] [Jacks 68].

The quality of user interfaces often determines whether users enjoy or despise a system, whether the designer of the system are praised or damned, whether a system succeeds or fails in the market. Indeed, in such critical applications as air-traffic controlling and nuclear-power-plant monitoring, a poor user interface can contribute to accidents of catastrophic proportions [Foley 90].

In the earliest interactive computers, the user interface was the relatively simple, but often awkward, textual input and output. Now, graphical interfaces have replaced textual interfaces as the standard means for user-computer interactions. It is generally agreed by researchers in human factors, interface design and the new field of scientific visualization that human are more adept at processing pictorial information than numerical or textual information [IGI 89]. In today's graphical user interfaces, communication is made richer by more powerful computers, new input and output devices, and sophisticated software. For example, many constant elements of appearance and interaction have become embodied in commercial user-interface products providing basic window management for a variety of applications. In the last few years, three dimensional user interfaces have been explored, and new strategies, such as fisheye views, have been proposed.

1.4 Thesis Goal

Techniques for displaying large hierarchical networks are an important part of the Intelligent Graphic Interface (IGI) project³, whose goal is “to combine advanced computer graphics technology with expert systems and human factors engineering to produce an Intelligent Graphic Interface”, essentially for real time supervisory and control systems [IGI 90]. In a 1989 feasibility study, the IGI team designed a graphical representation of the telephone network used in BC Tel, and performed human factors experiments to compare textual and graphical displays [IGI 89]. The focus of the experiments was on problem identification and solution determination. Given a scenario, each subject was asked to identify if a problem situation exists; if so, the subject then evaluated the problem with respect to available information and finally determined the correct solution. The results showed that the naive user group did better with graphical presentation, and the expert user group did better with textual presentation. The feasibility study report of the IGI project suggested that a new and more effective technique for graphical presentation, such as fisheye views, should be explored [IGI 90]. The fisheye view, pioneered by G.W. Furnas, is a relative new technique. It is “based on an analogy to a very wide angle, or ‘fisheye’, lens. Such a lens can show places nearby in great detail while still showing the whole world – simply by showing the more remote regions in successively less detail.” [Furnas 86].

³IGI is a large (five-year, \$6.8 million) joint university-industry project in which the work represented by this thesis plays a significant role.

Based on previous research in the IGI project, this thesis develops a new technique, called **variable zoom**, to represent large networks by combining hierarchical representations with a generalized fisheye view. A preliminary human factors experiment showed that the variable zoom technique provided a significant advantage over a conventional display method [IGI 92-0061]. This new technique will be integrated into two experimental industrial prototypes of the IGI project in the near future [IGI 92-0046] [IGI 92-0087].

To summarize, the main goal of this thesis is:

after investigating current technologies for representing large information spaces, to develop a new and more effective technique to display large hierarchical networks.

We do not yet have an existing comprehensive model for representing large hierarchical networks. Thus, in designing any new technique, we must make educated “guesses” as to the layouts should look like, and then test the results through human factors experiments. A second goal of this thesis is to work with a human factors group to test the method developed. We note that such testing is an expensive lengthy procedure requiring close collaboration between human factors experts and the designer of the new technique.

In the variable zoom method, the layouts of large hierarchical networks are designed to integrate details of interest with levels of context in a single window: the details are uniformly magnified while the contexts represented by icons are gradually demagnified. The magnification/demagnification factors are adjusted according to

the environment and the user's requests. To test the variable zoom layouts, a preliminary human factors experiment has been performed [IGI 92-0061] which showed that these layouts provided a significant aid to overcome the memory and keyhole problems encountered in the traditional method mentioned in Section 1.2. As a part of the IGI industrial prototypes, the variable zoom method will be tested by further human factors experiments [IGI 92-0087].

The remainder of this thesis is organized as follows. Chapter 2 introduces and evaluates current technologies for displaying large information spaces. Chapter 3 develops the fundamental ideas and algorithm for the variable zoom technique. Chapter 4 describes a human factors experiment which compared the variable zoom with a traditional technique. Chapter 5 compares the variable zoom method with other research, and gives some suggestions for future work. Finally, Chapter 6 provides a summary.

CHAPTER 2

LITERATURE REVIEW

Many methods have been developed to display large 2D (two-dimensional) information spaces on computers. They can be divided into three classes: (1) traditional pan/zoom methods; (2) map views (often refer to as “you are here”); and (3) distorted view methods, which currently emphasize various fisheye strategies. This chapter first describes and evaluates these methods with the taxonomy illustrated in Figure 2.1. This taxonomy focuses on the fisheye view (distorted view) method, the foundation of our new technique. Then we review some recent research about hierarchical representations to help the user understand the variable zoom method. Lastly, we outline our new technique which solves some of the problems of the earlier methods.

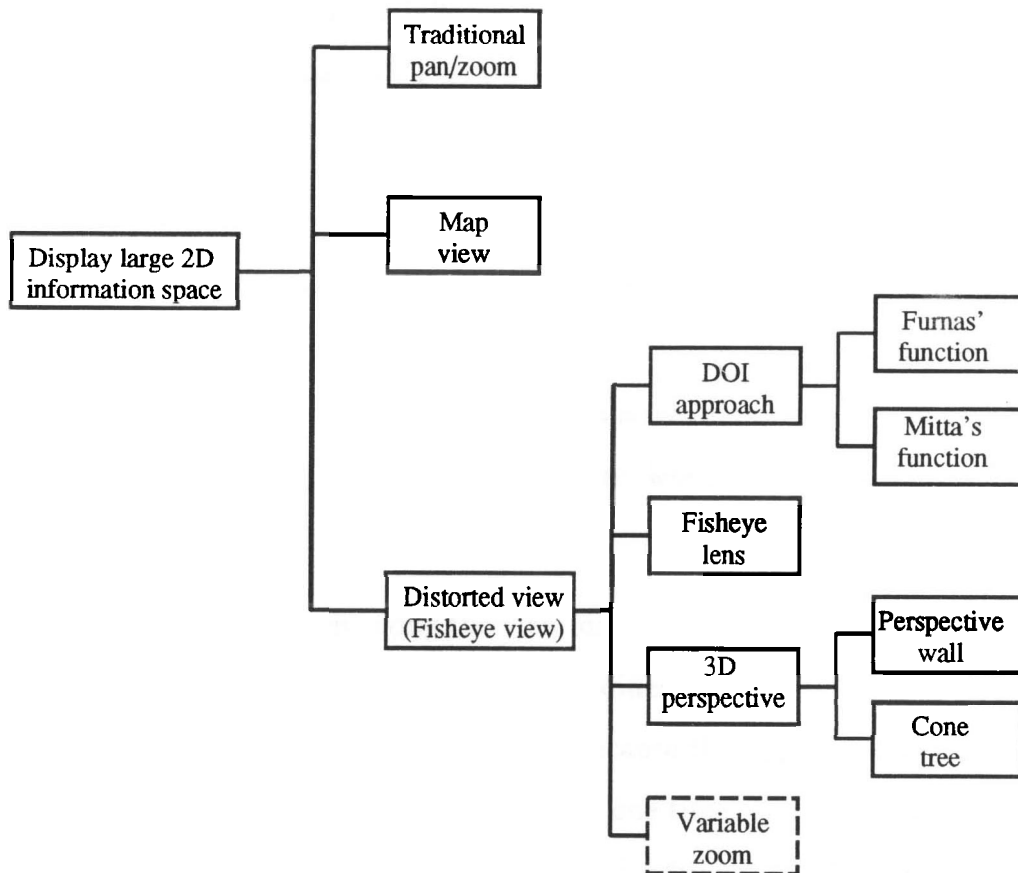


Figure 2.1: The taxonomy of methods for displaying large 2D information spaces. (The dashed rectangle indicates the position of our variable zoom method.) .

2.1 Traditional Method and Navigation Problem

The traditional pan/zoom method is widely used in many computer applications, such as *IBM PC Microsoft Windows* and *Apple Macintosh MacDraw*. The user can pan and scroll a window horizontally or vertically over a large information space with a cursor and scrolling bars. He can also either view details by uniformly scaling up the entire space, or the global structure by uniformly scaling it down.

The traditional method is simple and intuitive. However, for large information spaces, it suffers from severe **navigation** problems: the user easily gets lost, i.e., finds himself in some wrong place with little idea how to get to the right one [Furnas 86]. Researchers have found that browsing a large information space with a pan/zoom method tends to obscure the global structure; when we zoom in to see the detail, the context is lost [Henry 91].

Here, navigation is changing the view location and size so that the user can look at different parts of the information space at different scales. Navigation becomes very important for a large information space, because the display medium is too small to show the entire image in sufficient detail at one time. In recent years, the navigation and detail vs context problems arising from traditional methods have received much attention [Furnas 86] [Beard 90] [Mackinlay 91] [Fairchild 88]. Several approaches were proposed basically along two directions: map views and fisheye views. The former displays the detail and context in two or more windows simultaneously, while the latter distorts the view such that detail and context can be integrated in a single window. They are described in detail in the following two sections.

2.2 Map View Method

The map view method displays an information space in two (or more) windows, referred to as **map view** and **main display**. The map view provides a “miniature” of the entire information space with a “you are here” indicator. Usually this indicator is a wire-frame rectangle, which can be moved and resized, to indicate the location of the zoomed part which is displayed in detail in the main display. Figure 2.2 illustrates an example used in [Beard 90]. The map view method is also used in many other systems, such as [Brooks 86], [Delisle 86], [Smith 86], and [Halasz 87]. Variations of the map view are often found in many software packages, such as Lotus *Manuscript* and VLSI design tools such as *ICARUS*.

Recursive map views can be used to display multiple levels of context for very large information spaces. A detail view can become a map view with a rectangle showing the location of still another detail view [Donelson 78]. “This approach greatly increases the richness of the information space.” [Beard 90].

The map view method has the advantages of (a) viewing both of local detail and global context simultaneously, and (b) displaying the detail area uniformly to help the user easily understand local information. However, it has three major drawbacks. First, it forces the viewer to mentally integrate the detail and context together to form a single view. Second, the map window requires extra space and, sometimes, is difficult to arrange. For example, in Figure 2.2, if we put the map window onto the main display, some important details in the main display could be obscured; if we put it outside the main display, some of the rectangular screen space could be wasted. Third, we might wish to display two or more detail areas and their connections

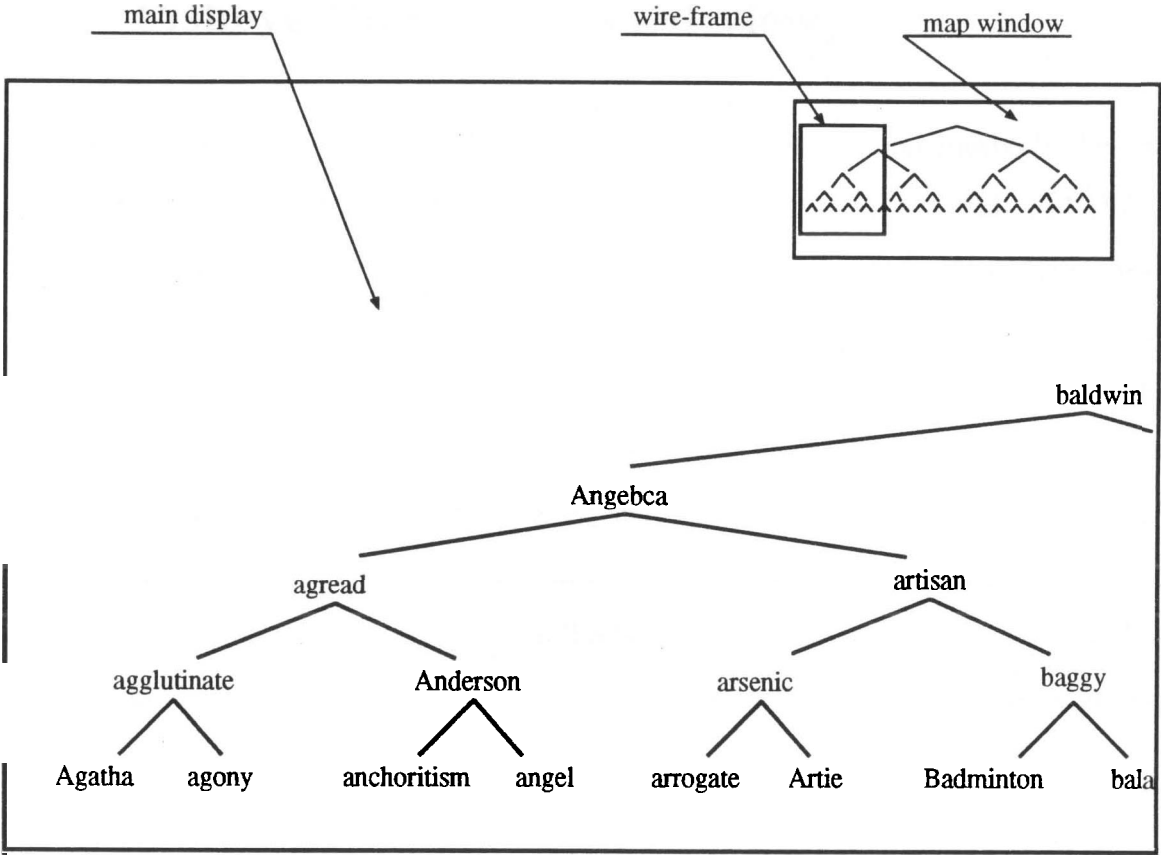


Figure 2.2: Using the map view method to display a large binary tree [Beard 90]

simultaneously. While it might not be difficult to extend this method to view multiple details at same time, it is extremely difficult to show their connections directly when these details are not close enough.

2.3 Fisheye View (Distorted view)

To overcome the navigation problems encountered by traditional methods, Furnas proposed generalized fisheye views “to provide a balance of local detail and global context” in a single window [Furnas 86]. The name *fish-eye* comes from optical *fish-eye lens*, a very wide angle lens showing detail near the optical axis of the lens while compressing regions far from the axis. Furnas gave many compelling arguments indicating that fisheye views might be ubiquitous in human processing of large information structures [Furnas 86]. Resnikoff observed that the human eye and other biological systems process vast amounts of information by integrating a focused view for detail with a general view for context [Resnikoff 89]. In the **following three subsections**, we describe three different approaches to fisheye views: DOI (the Degree Of Interest), fisheye lens, and 3D (three-dimensional) perspectives.

2.3.1 DOI (Degree of Interest) Approach

Fisheye views display details near a focal point and only important landmarks further away. In the DOI approach, fisheye views are generated by computing DOI values for every object, then displaying only those objects with DOI values greater than a threshold.

In Furnas' DOI function, the DOI increases with *a priori* importance and decreases with distance.

$$DOI(x | y) = API(x) - D(x, y) \quad (2.1)$$

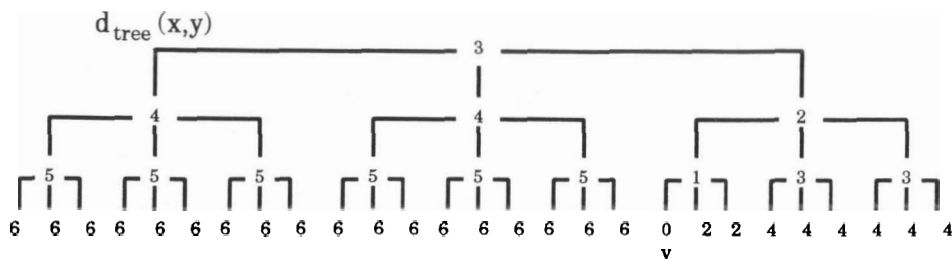
where $DOI(x | y)$ is the user degree of interest in point x , with current focal point y , $API(x)$ is the global *a priori* importance of x and $D(x, y)$ is the distance from x to y .

Figure 2.3 illustrates an example of applying Equation (2.1) on a tree structure [Furnas 86]. In Figure 2.3 (a), $D(x, y) = d_{tree}(x, y)$, the distance between x and y in the tree. In Figure 2.3 (b), $API(x) = -d_{tree}(x, root)$, the distance of x from the root, under the assumption that points closer to the root are intrinsically more important. Combining (a) and (b) in Figure 2.3 (c),

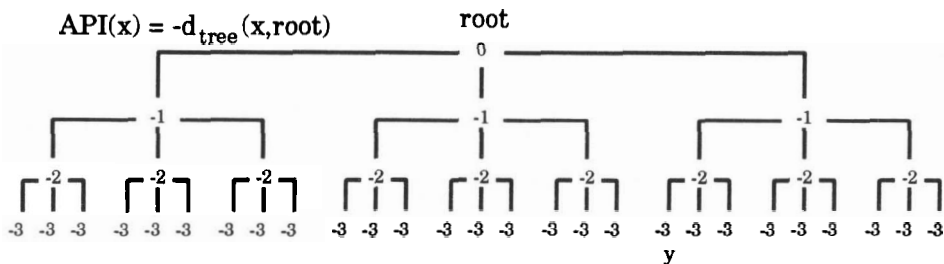
$$DOI_{tree}(x | y) = API(x) - D(x, y) = -(d_{tree}(x, y) + d_{tree}(x, root))$$

An arithmetically larger DOI value means the corresponding point is more interesting. Therefore, the points with $DOI = -3$ form the most interesting subset, those with $DOI = -5$ form the next most interesting subset, etc. One can obtain fisheye views by choosing a threshold, k , and only displaying those points with $DOI(x) \geq k$. For example, letting $k = -3$ displays only the most interesting subset; $k = -5$ forms the view with the most interesting and next most interesting subsets, etc.

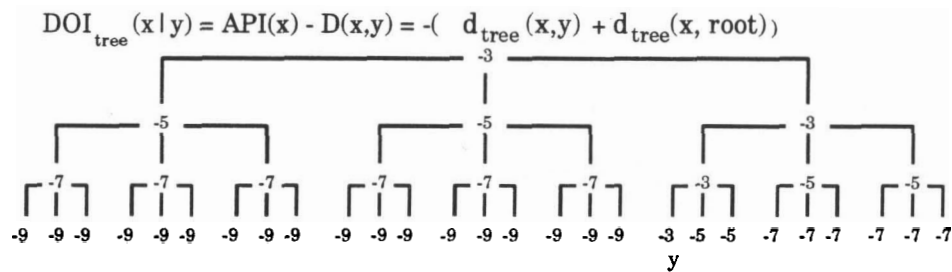
Furnas' DOI function, Equation (2.1), is especially suitable to information organized as a tree structure. Mitta extended this formula to the information described by a network [Mitta 89], and made multiple foci possible.



(a) Distance from y: $D(x,y) = d_{tree}(x,y)$.



(b) A Priori Importance in the tree, given by $API(x) = -d_{tree}(x,root)$.



(c) The fisheye DOI.

Figure 2.3: DOI function for a tree structure information space [Furnas 86]. (a) distance; (b) *A Priori* importance; and (c) the fisheye DOI. Notice that this is not the physical layout of the information space, but the abstract data structure. For example, “C” source code can be expressed as this kind of tree data structure. .

For a network with multiple focal points, Mitta's DOI function is

$$DOI(x_i) = API(x_i) - \sum_j D(x_i, y_j) \quad (2.2)$$

where, for a node x_i , $DOI(x_i)$ and $API(x_i)$ are the degree of interest and the global importance rating respectively; and $D(x_i, y_j)$ is the minimum path distance between x_i and a focal point y_j . Mitta used this function in IMIS, the Integrated Maintenance Information System, to aid in tasks associated with aircraft maintenance [Mitta 89].

In the DOI approach, two points are especially useful for displaying hierarchically organized networks: (a) In a hierarchy, objects closer to the root are intrinsically more important [Furnas 86]; (b) the user interest in an object depends on its position in the abstract data structure of the information space (e.g. which level and how far from the focal point), rather than the geometric position on the physical layout of the information space, such as that used in a 3D perspective method.

Though several variations of the DOI function have been suggested [Furnas 86] [Mitta 89], the DOI function for a general data structure has not yet been found. The more important problem arising from the DOI approach is that thresholding the DOI values causes the visualization to have gaps. Such gaps could make navigation difficult, because the desired destination might be in one of the gaps. Also, gaps might cause confusion while transiting from one view to another as familiar parts of the visualization suddenly disappear into gaps [Mackinlay 91].

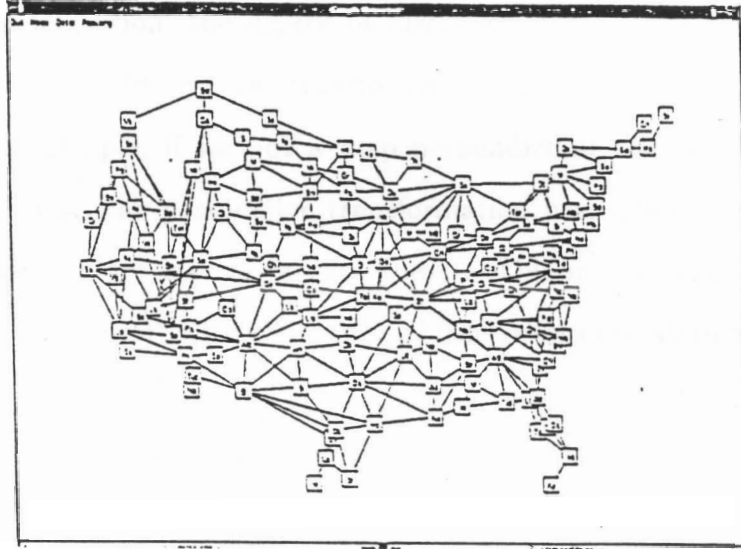
2.3.2 Fisheye Lens Transformation

In [Sarkar 92], the authors described a system for viewing and browsing planar graphs using a software analog of a optical fisheye lens. A normal layout is geometrically transformed into a fisheye view. The positions, sizes, and levels of detail of objects displayed are computed based on user-specified functions of an object's distance from the focus and the object's preassigned importance in the global structure.

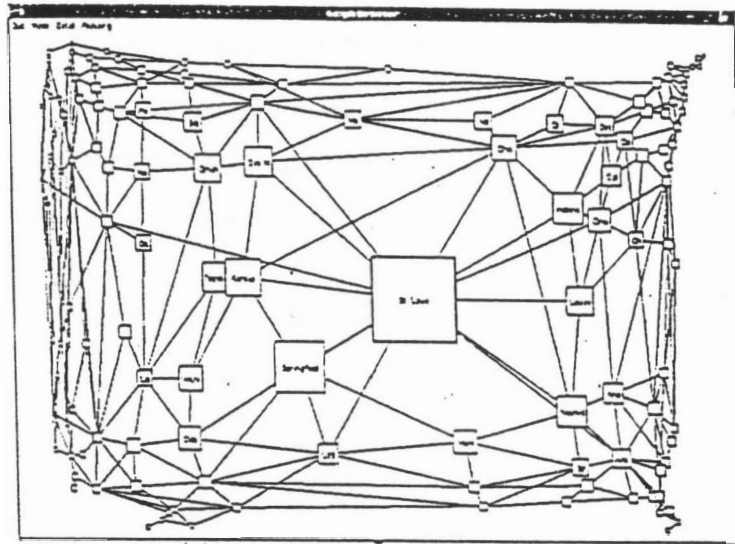
A graph mapping the major cities in United States and the major highways between them was used as a typical example in [Sarkar 92], as shown in Figure 2.4(a). For each vertex, Sarkar and Brown computed the fisheye display using four functions: the position, the size, the amount of detail to be shown and the visual worth (some functions had their own sub-functions); here the visual worth of vertex v depends on the distance between v and the focus, and on v 's API. Figure 2.4(b) shows one of the results.

In principle, as pointed out by Sarkar et al, the fisheye lens transformation can be applied to an information space with an arbitrary structure by changing the interpretation of "distance", and it can also be extended to multiple foci. But the main problem is that, to the user, the results from fisheye lens transformations "seemed somewhat unnatural, especially when applied to familiar objects, such as maps". In addition, the fisheye lens transformation "does not preserve parallelism between edges" [Sarkar 92].

To understand the problems resulting from the fisheye lens transformation, we compare the transformation with a 3D perspective transformation which is closer to human visual model. There are at least two important differences. First, in the



(a) A normal layout before a fisheye lens transformation.



(b) The fisheye view after the fisheye lens transformation.

Figure 2.4: An example of fisheye lens transformation [Sarkar 92].

perspective transformation, the degree of distortion depends on the distance from the **viewer**, but in the fisheye lens transformation, it depends on the distance from the **focus**. For example, if we put a map perpendicular to the viewing direction and far enough away, the perspective transformation basically does not distort the map, but the fisheye lens transformation does¹. This might explain why the user feels “somewhat unnatural” with the results of fisheye lens transformation. Second, a straight line remains a straight line after a perspective transformation, but, in general, becomes a curve after a fisheye lens transformation.

2.3.3 3D Perspective Projection

This is a relatively new approach using a 3D perspective method to view a 1D or 2D information space in order to obtain a fisheye effect. “A balance of local detail and global context arises automatically from the geometry of point perspective in three dimensions: a few nearby points loom large and those further away appear smaller and smaller.” [Fairchild 88]. Here, we outline two of those methods: **Perspective Wall** [Mackinlay 91] and **Cone Tree** [Robertson 91b].

The Perspective Wall method is for visualizing **linear**² information by smoothly

¹Consider a $40\text{cm} \times 40\text{cm}$ map, perpendicular to the line of sight and 80cm from the viewer. The ratio of the areas of two nodes, $2\text{cm} \times 2\text{cm}$, one at the center, another at the corner, is 1:0.89, after the perspective transformation. But it becomes 1:0.03, after the fisheye lens transformation (the user-specified constants [Sarkar 92] used here are the same as in Figure 2.4(b))

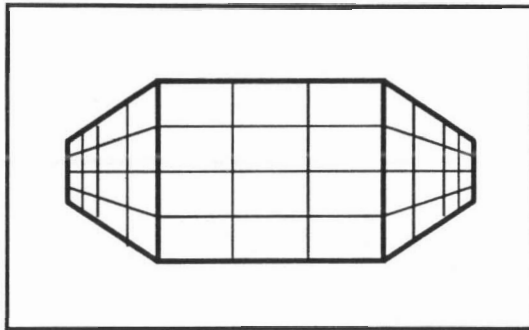
²[Mackinlay 91] used “linear” to refer to a 2D layout with wide aspect ratio. Sometimes such a 2D layout is called a one-dimensional structure, through which the user navigates to locate the desired place [Beard 90].

integrating detailed and contextual views. It folds wide 2D layouts into a pseudo 3D visualization that has a center panel for detail and two perspective panels for context, as shown in Figure 2.5 [Mackinlay 91]. It makes the immediate neighborhood of the detailed view larger than more distant parts of the contextual view. When the user selects an item, the wall moves this item to the center panel with a smooth animation.

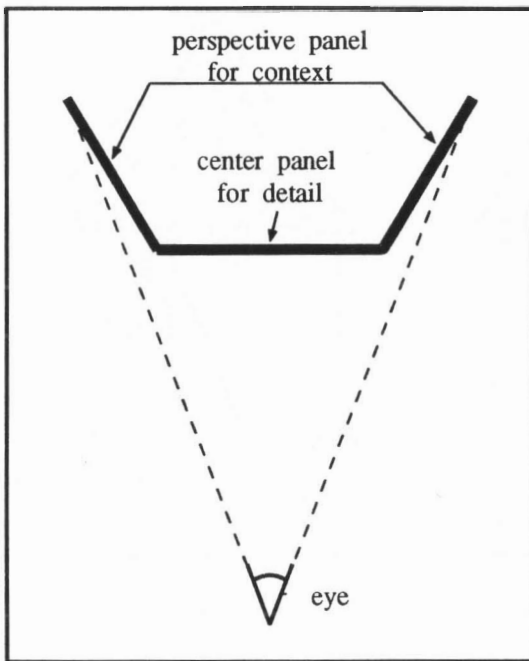
Two principles used in the Perspective Wall are important to us: (a) uniform detail + distorted context; and (b) gradually distorting contexts from near to far. The major disadvantages are: (a) it is limited to linear information spaces, and difficult to extend to a 2D network structure; (b) multiple foci are not allowed; and (c) use of computer screen area is inefficient, because the trapezoidal perspective panels cannot fit in the rectangular screen without wasted spaces.

Cone Trees provide fisheye views with 3D perspective and coloring [Robertson 91b]. As shown in Figure 2.6, Cone Trees are hierarchies laid out uniformly in three dimensions. The top of the hierarchy is placed near the ceiling of the room, and is the apex of a cone with its children spaced evenly around the perimeter of its base. The next layer of nodes is drawn below the first, with their children in cones. The body of each cone is shaded transparently, so that the cone is easily perceived yet does not block the view of cones behind it. When a node is selected, the Cone Tree rotates so that the selected node and each node in the path from the selected node up to the top are brought to the front and highlighted.

It is difficult to represent a hierarchical network with a Cone Tree, because there are not only links between parent-nodes and child-nodes, but many links between nodes at the same level.



(a) User view (front view)



(b) Basic idea (top view)

Figure 2.5: Sketch of a Perspective Wall [Mackinlay 91].

(a) User's view on the screen. (b) Top view showing the basic idea of Perspective Wall (bold line).

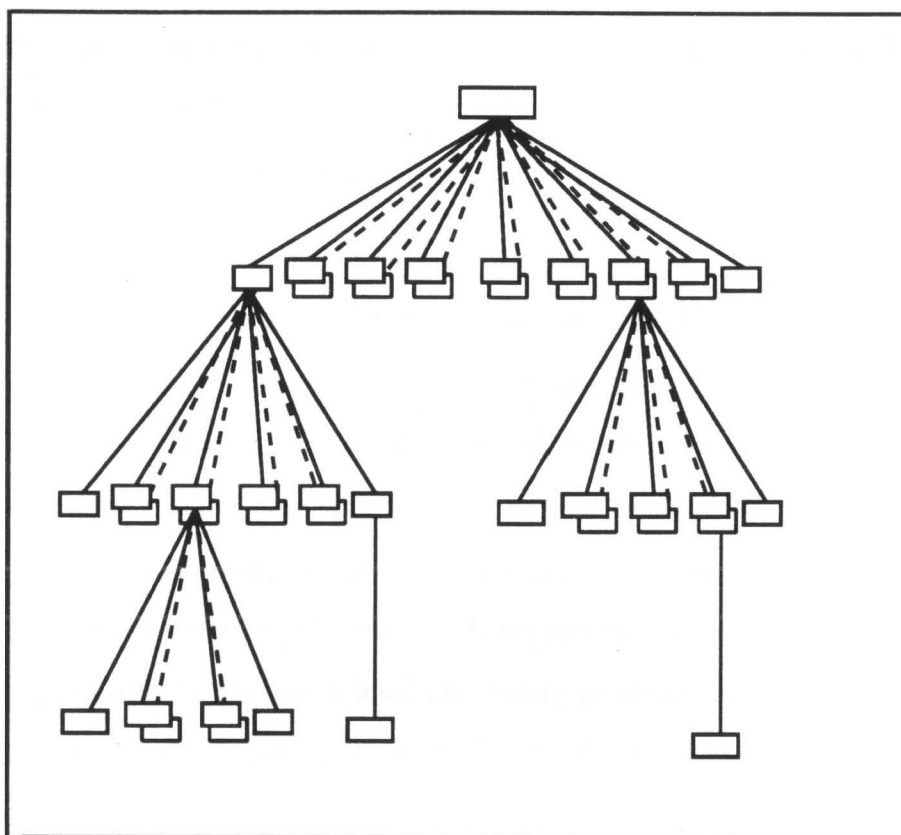


Figure 2.6: Sketch of a Cone Tree [Robertson 91b].

2.4 Hierarchical Abstraction and Representation

Hierarchies are almost ubiquitous, appearing in many different applications. Examples include structured programming languages (e.g. LISP, PASCAL and C), hierarchically organized text (e.g. manual, legal codes), various highly structured science and technological knowledge domains (e.g. biological taxonomies), and hierarchical file systems (e.g. UNIX). Here, we only review representative recent papers related to the display of large 2D information space.

[Card 91] described some general observations in information processing. They found that lower levels of an information processing system simplify and organize information supplying higher centers with aggregated forms of information through abstraction and selective omission. Even when information is on the computer screen, there is a generic problem that the volume of information to be processed is large relative to the ability of the user. An answer is to stage processing by recoding the information in progressively more abstract and simpler representations. The abstractions produced by the lower-level processing predetermine, to a considerable extent, the patterned structures that the higher-level processing can detect [Resnikoff 89]. The higher-level processing, in turn, reduces still further the quantity of information by processing it into yet more abstract and universal forms.

[Fairchild 88] proposed a system, called SemNet, to represent large knowledge bases with 3D graphics³. It introduced new cluster-objects representing sets of objects,

³Because the photographs of [Fairchild 88] which illustrated the layouts of SemNet were not clear, accurate evaluation of the method is difficult.

and assigned greater importance to these cluster-objects than to individual elements [Fairchild 88]. They also suggested that we could use a special icon carrying some identifying information to represent a cluster-object, to help the user deal with the fisheye distortion. This information might be an indicator, a label, the cluster-name, an example inside the cluster etc.

2.5 Variable Zoom: A New Technique

To overcome navigation problems encountered in traditional methods, researchers have developed a variety of methods to improve the displays of large 2D information spaces. But there is still no ideal method to display the large hierarchical networks considered in our project.

This thesis develops a new technique, called **variable zoom**, to display these networks. Its main features are:

- a tight marriage of fisheye views with hierarchical representations;
- uniformly magnifying details;
- gradually demagnifying levels of contexts with increasing distance from the detailed view;
- adjusting magnification/demagnification factors, according to the environment and the user's requests; and
- allowing multiple foci.

A detailed description of the variable zoom technique is the subject of the next chapter. After presenting our new technique and its human factors experiment results, we will come back to compare the variable zoom technique with the previous research described in this chapter.

CHAPTER 3

VARIABLE ZOOM TECHNIQUE

This chapter describes the variable zoom (VZ) method in the following order: fundamental ideas, conventions and definitions used in this chapter, important parameters, the algorithm for basic-operation, and the varying balance factor method.

Two abstract networks, one “smaller” and the other “larger”, are alternatively used as examples in this chapter. The smaller one, which includes 11 nodes and can be considered as a portion of a large network, is used to illustrate: (1) a hierarchical organization and representation; and (2) the algorithm for the basic-operation. The larger one, a four-level network including about 700 nodes and 700 links, is constructed to (1) test the behaviour of the VZ technique in a large hierarchical network environment; and (2) illustrate the varying balance factor method. To help the reader distinguish them, we use uppercase letters as the nodes’ names for the first one, and lowercase letters for the second.

3.1 Fundamental Ideas

3.1.1 Hierarchical Organization and Representation

As mentioned in Section 1.1 and 2.4, people often organize a large network in a hierarchical structure. For example, Figure 3.1(a) is the normal layout of a network of 11 nodes; (b) clusters the network into four clusters; (c) uses icons to represent clusters to obtain a simple higher level network; and (d), in turn, represents the entire network with single icon at yet a higher level.

In Figure 3.1, circles represent “leaf” nodes (sometimes referred to as **physical nodes**), the lowest level of detail, such as switches in a telephone network or breakers in a power grid. Rectangles are the **logical nodes**, which represent lower level networks and could be considered as substations in a power grid. Solid links connecting physical nodes represent **physical links**, such as circuits connecting switches. A dashed line, connecting logical nodes or one logical node with one physical node, is a **logical link**, which represents one or more lower level links (logical or physical). The concepts of physical and logical nodes and links are important to the user, as described in the human factors experiment (Chapter 4).

Here, hierarchical structure and logical (iconic) presentation are bound together. For example, ‘PORTION’ in Figure 3.1(d) is a node in a higher level network (the rest of the network is ignored here), and it is also an icon representing its sub-network which includes ‘A’ to ‘D’. In turn, the nodes ‘A’ to ‘D’ of the network shown in Figure 3.1(c), are iconic representations of their lower (bottom) level sub-networks. This hierarchical abstraction allows us to easily omit the lower level detail when not

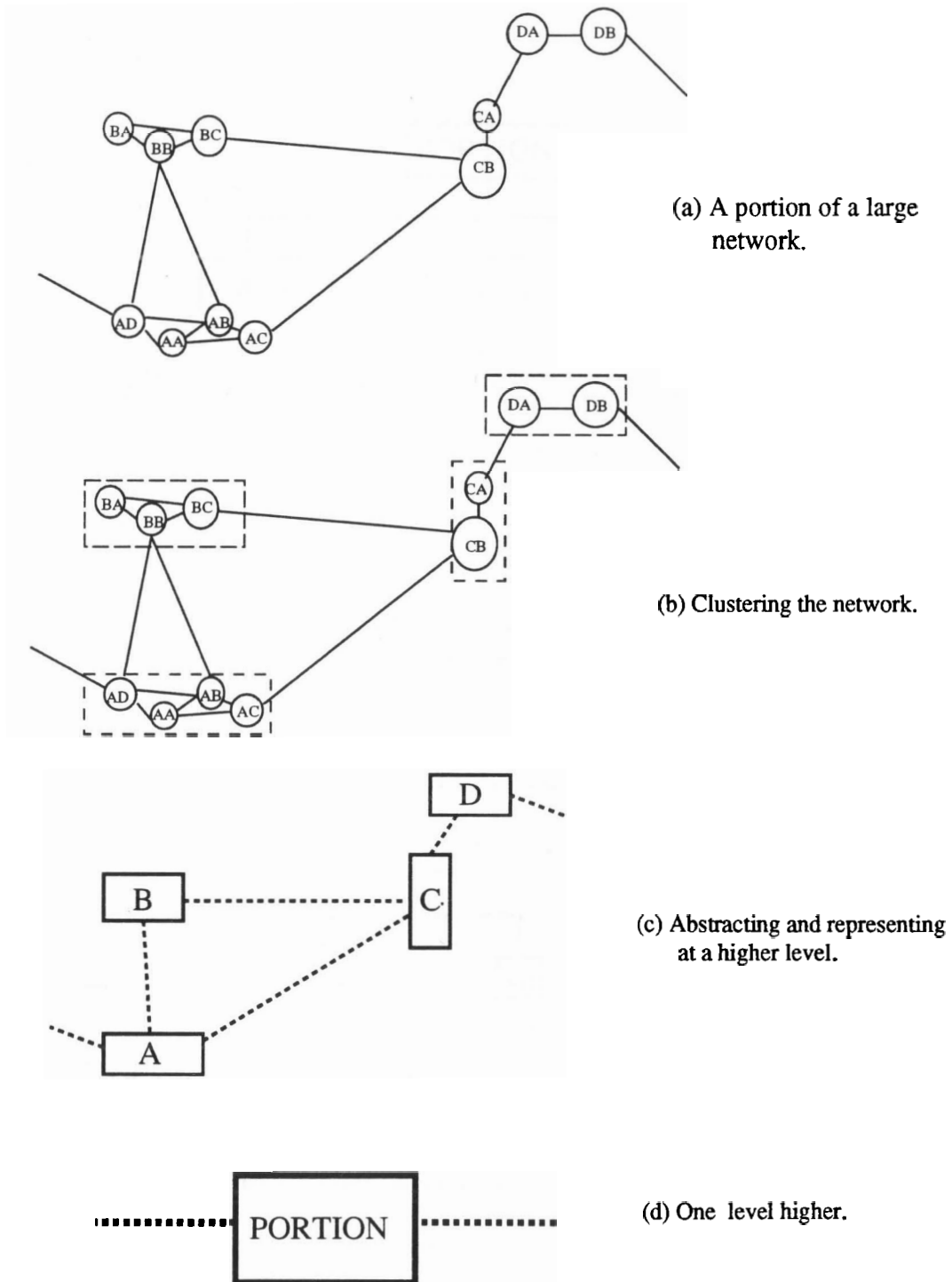
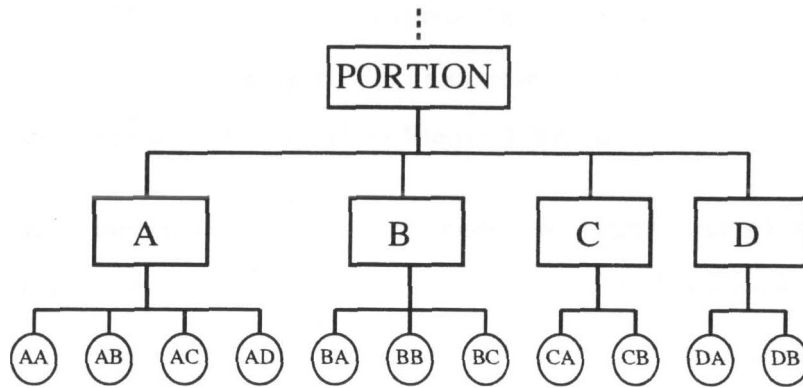
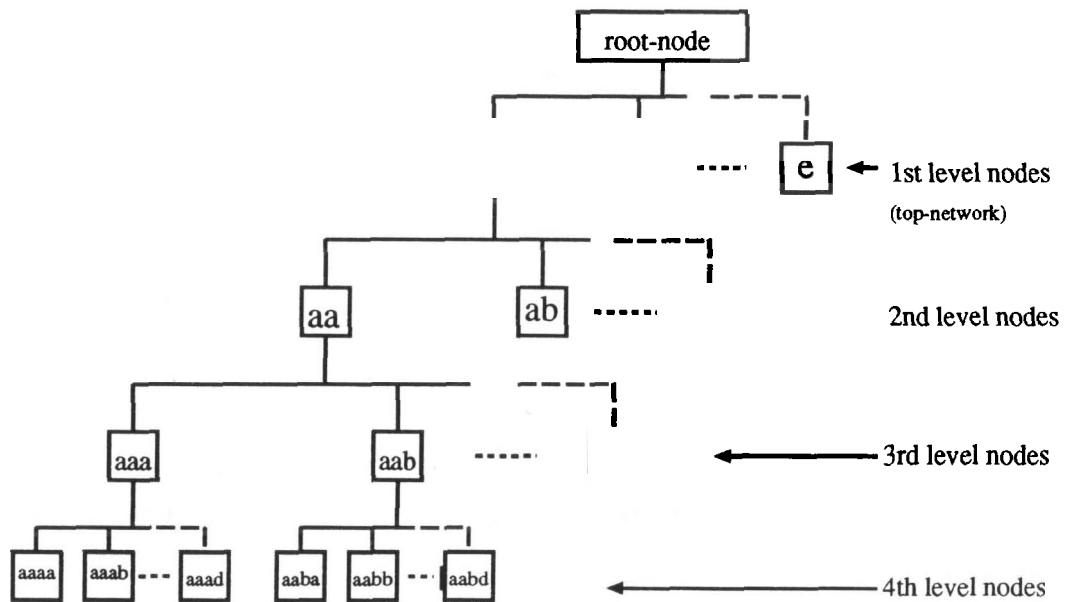


Figure 3.1: A hierarchical organization and representation for a network.



(a) The hierarchical structure for the nodes in the smaller network shown in Figure 3.1



(b) The hierarchical structure for the nodes in the larger network shown in Figure 3.4

Figure 3.2: Tree representation of hierarchical nodes.

of current interest, and to greatly simplify the higher level display.

Often we use tree structures to express the hierarchical relations between the nodes of such networks, ignoring all the links between nodes. Figure 3.2(a) is the tree structure of the smaller network, while Figure 3.2(b) represents the larger one.

As long as a hierarchy is maintained, nodes in a large network can be clustered in any way the designer wishes, e.g. by using geographical distance, by task-specific relationships between nodes etc. The variable zoom technique focuses on how to display a large network which has been organized into a hierarchical structure, rather than on how to organize it into a hierarchical structure.

3.1.2 Fisheye View Strategy

One key point to obtaining the fisheye effect is to uniformly magnify appropriate parts of the lower level in order to show details, and to embed the details into the rest of the network which is scaled down.

Figure 3.3 illustrates a 1D (one-dimensional) view of the VZ approach for a single level. The VZ method generates a fisheye view by magnifying enlarging segments by an **enlarging factor** F_e and demagnifying shrinking segments by a **shrinking factor** F_s . The position of all nodes must be recomputed to keep the entire network within the space of its parent node while allocating more space for enlarging segments.

F_e and F_s are two important parameters of the VZ method and their values are dependent on both the entire environment and the user's requirements (i.e. which nodes to look at in detail). For this reason we call this technique a **variable** zoom

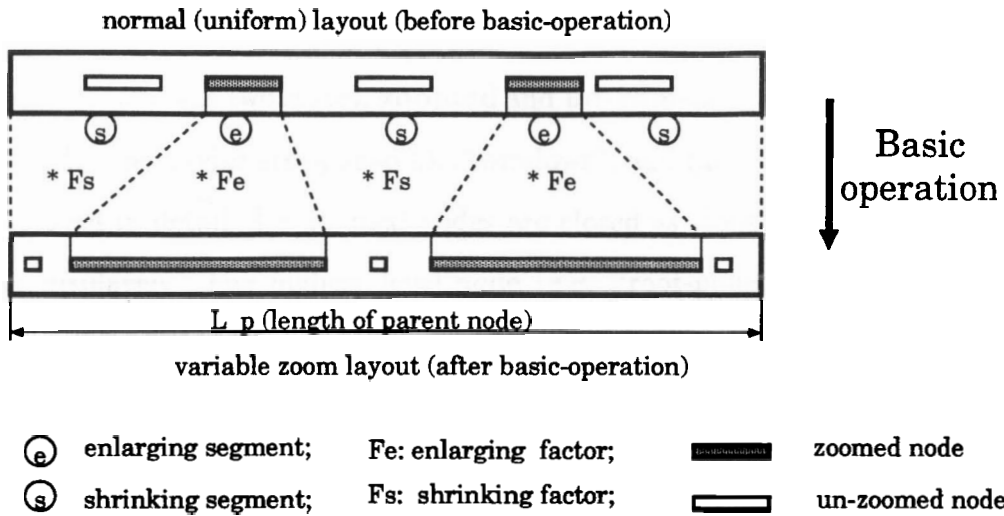


Figure 3.3: One-dimensional view of the variable zoom layout.

method. The calculations of F_e and F_s are described in Section 3.3 in detail.

Another important part of producing a fisheye view is gradually distorting context objects according to their distances from the detailed view. In the VZ method, the size of nodes in a higher level context decreases with their distance from the detailed view by varying the **balance factor** K_b , as described in Section 3.5.

3.2 Conventions and Definitions

3.2.1 Conventions

In our algorithm, nodes are assumed to be rectangles (including squares), and the positions and sizes of all nodes at all levels are known. In general, non-rectangular shapes such as circles, triangles and symbols, could be included in bounding rectangles

in order to deal with non-rectangular shapes.

A logical node has two states, **zoomed** and **un-zoomed**, which can be changed by the user. Zoomed nodes are opened like “windows”, and their immediate sub-networks are displayed in detail. Un-zoomed nodes are closed as icons and their sub-networks are not displayed. The highest level node (e.g. ‘root-node’ in Figure 3.2(b)) is a special node, which is always in a zoomed state as the start point of the variable zoom method. The lowest level nodes (physical nodes) are always un-zoomed.

A network consists of nodes and links. In this chapter, a link is simply defined as a straight line connecting the centers of two nodes. A node-based algorithm is used here, because it is easy to display a link after the positions of its two end nodes have been determined. We leave the link-based algorithm in future work.

In this thesis, we derive algorithms basically from a one-dimensional point of view, and leave a two-dimensional derivation for future research. Algorithms are described only in the x direction, if the y direction is identical. A simple example often follows an algorithm to help the reader understand it.

In the definition of parameters, we use subscripts to indicate whether a parameter applies to the x direction or y direction, if necessary. For example $F_{e,x}$ and $F_{e,y}$ are the enlarging factors in the x and y directions respectively. We also use a prime to denote a parameter value after a basic-operation. For example, x_A and x'_A are the positions of node ‘A’ on the x-axis before and after a basic-operation. The concept of basic-operation is described in next subsection.

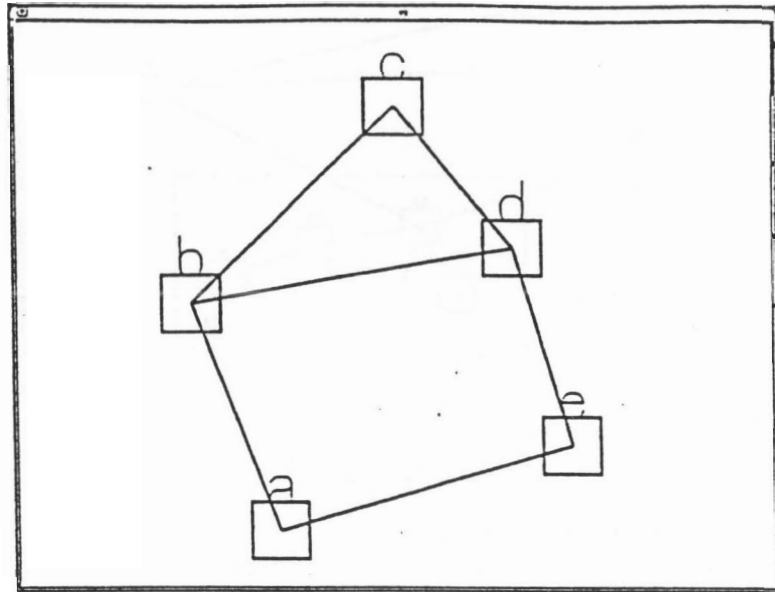
3.2.2 Basic-operation

We define a **basic-operation** as changing the state of one or more nodes, which share a common **Parent-node**, from an un-zoomed state to a zoomed state. For example, in Figure 3.4(a), we want to change the state of nodes 'a' and 'd' (both are in the top-network) from un-zoomed to zoomed, and keep other nodes of the top-network un-zoomed. Figure 3.4(b) shows the result: nodes 'a' and 'd' are magnified and their sub-networks 'aa' to 'ai' and 'da' to 'di' are displayed, and the rest of the top-network is demagnified as a context.

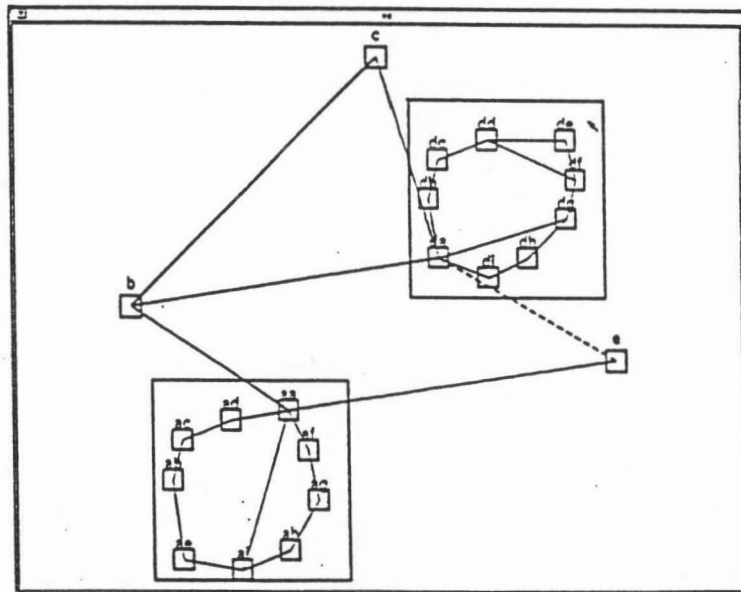
Three levels of nodes and two levels of network are involved in a basic-operation. The immediate children of a Parent-node ('root-node', in this example) are called **Current-nodes** (nodes 'a' to 'e'). Current-nodes and the links between them compose a **Current-network**. A lower level network represented by a Current-node is called a **Sub-network**, and its nodes are **Sub-nodes** ('aa' to 'ai' and 'da' to 'di'). The main points of a basic-operation are:

Before a basic-operation: The Parent-node has been in a zoomed state ('Root-node' is always zoomed in this example, as described earlier). All Current-nodes are in an un-zoomed state, and the Current-network is displayed at a uniform magnification. No Sub-network is displayed.

After the basic-operation: The Parent-node is unchanged. The zoomed Current-nodes ('a' and 'd') are magnified and display their Sub-networks; and the remainder of the Current-network is demagnified and retained within the Parent-node as a context. The Sub-networks of the zoomed Current-nodes are uniformly

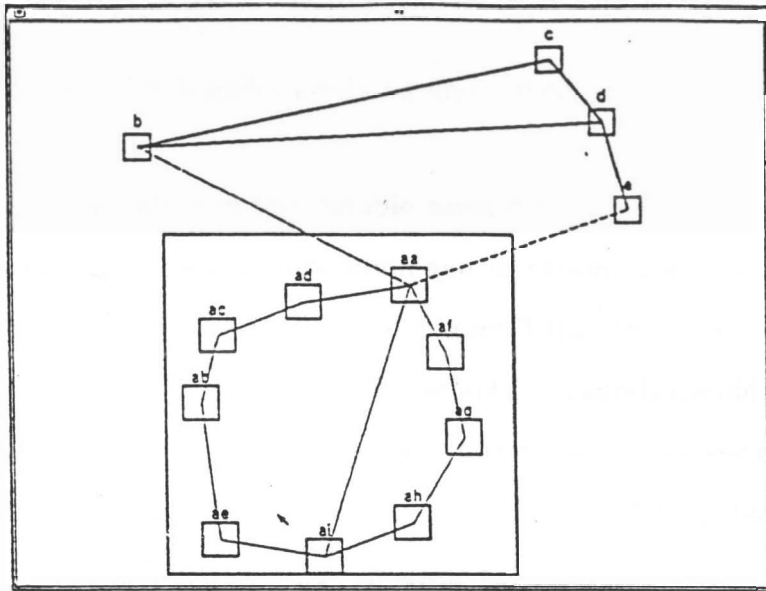


(a) Uniform layout of the top-network before basic-operation.

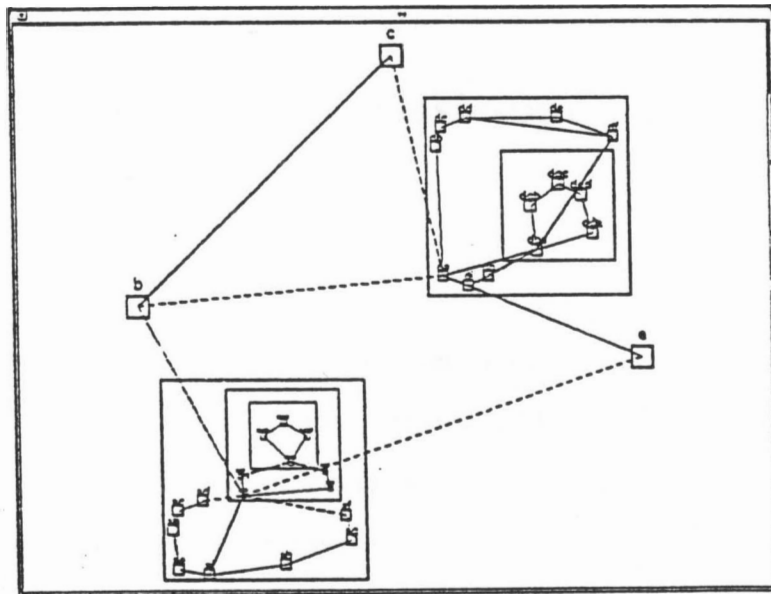


(b) After a basic-operation: nodes 'a' and 'd' are zoomed to show their Sub-networks; other parts of the top-network are shrunk as a context.

Figure 3.4: Examples showing the basic-operation.



(c) Another example of VZ layout. The magnification/demagnification factors are different from (b).



(d) An example involving multiple levels and multiple detailed views.

Figure 3.4: (continued).

displayed, and all Sub-nodes are in an un-zoomed state.

Any (complex) operation in the variable zoom method can be decomposed into a sequence of executing a new basic-operation and/or canceling an old basic-operation. For example, when we change node ‘d’ in Figure 3.4(b) from zoomed state to un-zoomed, retaining node ‘a’ zoomed, the VZ method first cancels the old basic-operation described above, and then executes a new basic-operation. In the new basic-operation, only node ‘a’ is zoomed as shown in Figure 3.4(c). Figure 3.4(d) shows that we can observe any level of network by recursively using the basic-operation. In the following two sections, we describe an algorithm for a basic-operation as well as the parameters used in it.

3.3 Parameters in the Basic-operation

3.3.1 S_a and S_z

Before a basic-operation, we project all Current-nodes on the x (or y) axis. If there is no overlap between these projections, such as in Figure 3.3, S_a is the sum of all projections, while S_z is the sum of the projections corresponding to the nodes to be zoomed. If there are overlaps between these projections, we will calculate S_a and S_z after merging the overlapping projections (segments). The accurate definition of **overlap** and procedure of **merge** are described later.

In some sense, S_a depends on the configuration of the Current-network and quantitatively reflects the environment, while S_z depends on which Current-nodes are to

be zoomed and reflects the user's requirement. S_a and S_z will be used to adjust F_e and F_s , after transforming them into ratios.

3.3.1.1 Calculating S_a and S_z

The algorithm calculating S_a and S_z can be divided into three main steps: (1) projecting relevant nodes onto the x-axis to form a projection segment list; (2) merging overlapping segments to form a new list without overlap; and (3) summing the segments in the new list.

ALGORITHM for S_a in the x direction.

1. Project **all nodes** onto the x-axis to form a projection segment list: $list_{a,x}$.
2. CHECK-MERGE segments in $list_{a,x}$ to obtain $list'_{a,x}$ (see CHECK-MERGE below).
3. $S_{a,x}$ is the sum of the lengths of all segments in $list'_{a,x}$.

ALGORITHM for S_z in the x direction.

1. Project **all nodes to be zoomed** onto the x-axis to form a list: $list_{z,x}$.
2. CHECK-MERGE segments in $list_{z,x}$ to obtain $list'_{z,x}$ (see CHECK-MERGE below).
3. $S_{z,x}$ is the sum of the lengths of all segments in $list'_{z,x}$.

CHECK-MERGE procedure:

1. Check each pair of segments in the list. If there is no overlap between any pair, CHECK-MERGE is finished and the list is the destination list $list'_{a,x}$ or $list'_{z,x}$.
2. If segment A **includes** B, delete B from the list.
3. If segment A and B **partially overlap**, add a new segment C on the list, and delete A and B. The left boundary of C is the minimum value of the left boundaries of A and B, and the right boundary of C is the maximum value of the right boundaries of A and B.
4. Repeat the above steps until all overlaps are removed.

Here, we divide overlap into inclusion and partial-overlap. For example, on the x-axis of Figure 3.5, segment $s_{x,A}$ includes segment $s_{x,B}$, and segment $s_{x,C}$ partially overlaps segment $s_{x,D}$. The precise definitions of inclusion and partial-overlap are: given segments A and B,

```

IF       $(L_A \leq L_B)$  AND  $(R_A \geq R_B)$ 
THEN   A includes B;
IF       $(L_B \leq L_A)$  AND  $(R_B \geq R_A)$ 
THEN   B includes A;
IF       $[(L_A < L_B)$  AND  $(R_A < R_B)]$  OR  $[(L_B < L_A)$  AND  $(R_B < R_A)]$ 
THEN   A and B partially overlap;

```

Here, L_A and R_A denote the left and right boundaries of the segment A, and L_B and R_B denote the left and right boundaries of the segment B.

3.3.1.2 Examples

Figure 3.4(a) is a simple example, in which all Current-nodes are squares and there is no overlap. S_a is equal to the sum of the lengths of all nodes (from ‘a’ to ‘e’), S_z is the sum of the lengths of the nodes to be zoomed (node ‘a’ and ‘d’), $S_{a,x} = S_{a,y}$, and $S_{z,x} = S_{z,y}$.

To help the reader understand the algorithm for a basic-operation and its parameters in detail, we redraw the network of Figure 3.1(c) in Figure 3.5 as the main example in this section and next section, in which the Current-network includes four overlapping rectangular nodes. Part A in Table 3.1 illustrates the procedure for calculating S_a and S_z .

3.3.2 Ratio Parameters

In the variable zoom algorithm, we often use the following ratios instead of the “absolute” parameters to make our discussion independent of the particular case.

$$r_1 \equiv \frac{L_p}{S_a} \quad (r_1 \geq 1) \quad (3.1)$$

$$r_2 \equiv \frac{S_a}{S_z} \quad (r_2 \geq 1) \quad (3.2)$$

$$r_3 \equiv \frac{S_z}{L_p} = \frac{1}{r_1 \cdot r_2} \quad (0 < r_3 \leq 1) \quad (3.3)$$

$$r_4 \equiv \frac{S_z}{S_a} = \frac{1}{r_2} \quad (0 < r_4 \leq 1) \quad (3.4)$$

Here, L_p is the length of the Parent-node. An example calculating these ratios is shown in Part B of Table 3.1.

Parameters	x-direction	y-direction
Part A: calculating S_a and S_z .		
$list_a$	$list_{a,x} = [S_{x,A}, S_{x,B}, S_{x,C}, S_{x,D}]$	$list_{a,y} = [S_{y,A}, S_{y,B}, S_{y,C}, S_{y,D}]$
$list'_a$	$list'_{a,x} = [S_{x,A}, S_{x,C+d}]$	$list'_{a,y} = [S_{y,A}, S_{y,C}, S_{y,D}]$
S_a	$S_{a,x} = 5.5$	$S_{a,y} = 4$
$list_z$	$list_{z,x} = [S_{x,B}, S_{x,C}]$	$list_{z,y} = [S_{y,B}, S_{y,C}]$
$list'_z$	$list'_{z,x} = [S_{x,B}, S_{x,C}]$	$list'_{z,y} = [S_{y,C}]$
S_z	$S_{z,x} = 3$	$S_{z,y} = 2$
Part B: calculating ratio parameters.		
L_p	$L_{p,x} = 12$	$L_{p,y} = 10$
$r_1 \equiv L_p/S_a$	$r_{1,x} = 2.18$	$r_{1,y} = 2.5$
$r_2 \equiv S_a/S_z$	$r_{2,x} = 1.83$	$r_{2,y} = 2$
$r_3 \equiv S_z/L_p$	$r_{3,x} = 0.25$	$r_{3,y} = 0.2$
Part C: calculating F_e and $F_{e,xy}$.		
$F_e = 0.5 \cdot r_1 \cdot (1 + r_2)$	$F_{e,x} = 3.09$	$F_{e,y} = 3.75$
$F_{e,xy} = \min(F_{e,x}, F_{e,y})$	3.09	3.09
Part D: calculating F_s and $F_{s,xy}$.		
$F_s = (1 - F_e \cdot r_3)/(1 - r_3)$	$F_{s,x} = 0.31$	$F_{s,y} = 0.48$
$F_{s,xy} = \min(F_{s,x}, F_{s,y})$	0.31	0.31

Table 3.1: Calculating parameters used in the variable zoom algorithm.
(Here, we set $K_b = 0.5$)

3.3.3 Enlarging Factor F_e

3.3.3.1 Derivation

To begin the derivation, we consider the ratio between the sum of enlarging segments and their immediate environment (the length of the Parent-node), in the VZ layout (refer to Figure 3.3). This ratio, R_z , is:

$$R_z \equiv \frac{F_e \cdot S_z}{L_p} = \frac{F_e}{r_1} \cdot r_4 \quad (0 < R_z \leq 1) \quad (3.5)$$

In this equation r_1 reflects, in some sense, the “environment” and r_4 reflects the user’s requests.

To satisfy the user’s request to zoom one or more nodes, R_z should meet the following three general requirements.

1. If even only one node is zoomed (since the size of the node may be arbitrarily small, we can think of $r_4 \rightarrow 0$), R_z should be bigger than some threshold value, set by the user, in order to make the Sub-network detail visible. Thus

$$R_z \geq \text{threshold} \quad (\text{when } r_4 \rightarrow 0) \quad (3.6)$$

2. If all nodes in the Current-network are zoomed in ($r_4 = 1$), then we should set $R_z = 1$, because it is unnecessary to retain any context.

$$R_z = 1 \quad (\text{when } r_4 = 1) \quad (3.7)$$

3. As the number of zoomed nodes increases from one to the total number of Current-nodes (in general as r_4 increases from 0 to 1), R_z should also increase from the threshold to 1 smoothly and monotonically.

A simple relationship satisfying the above three requirements is.

$$R_z = k_1 + k_2 \cdot r_4$$

where k_1 and k_2 are constants. From Equation (3.7), $k_2 = 1 - k_1$; and renaming k_1 as K_b ;

$$R_z = K_b + (1 - K_b) \cdot r_4 \quad (3.8)$$

Combining Equation (3.8), (3.5) and (3.4) we obtain F_e as,

$$F_e = K_b \cdot r_1 \cdot \left(\frac{1}{K_b} - 1 + r_2 \right) \quad (3.9)$$

A more general approach is to first design an R_z vs r_4 curve, which satisfies the above three general requirements and provides additional flexibility to the user, and then derive F_e equation from Equation (3.5). For example

$$R_z = K_b + (1 - K_b) \cdot (r_4)^u \quad (3.10)$$

where u is a positive parameter to control the behavior of the R_z vs r_4 curve. If $u = 1$, Equation (3.10) becomes the same as Equation (3.8); if $u > 1$, R_z increases slowly

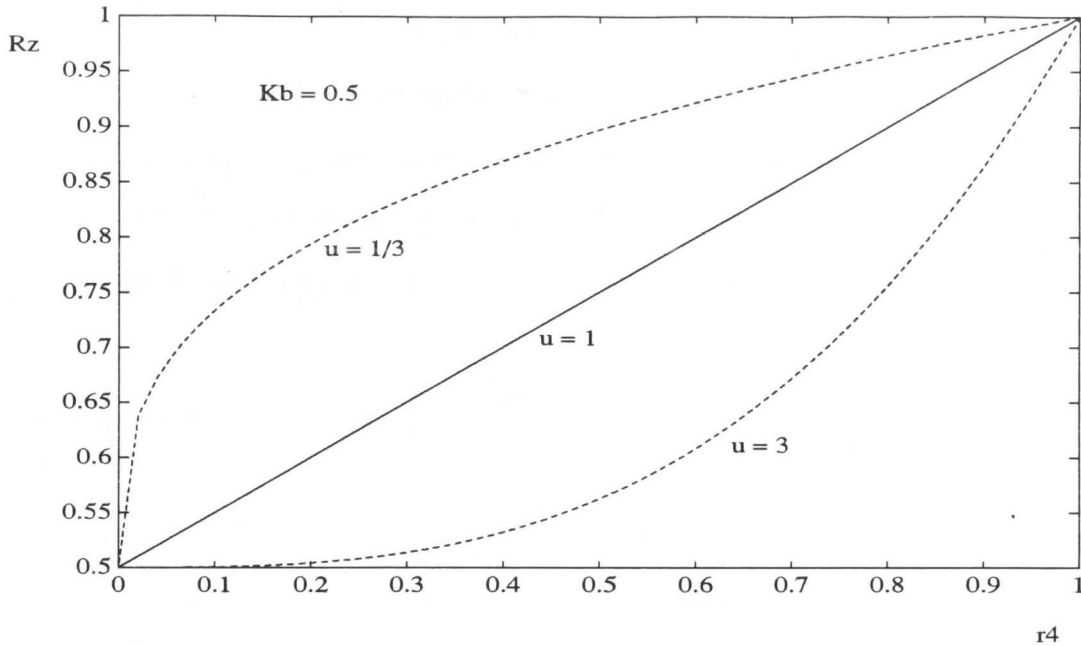


Figure 3.6: $R_z = K_b + (1 - K_b) \cdot (r_4)^u$.

at first and more quickly at the end with increasing r_4 ; and if $u < 1$, R_z increases quickly first and slowly last, as shown in Figure 3.6. From Equation (3.10), we can obtain F_e as:

$$F_e = r_1 \cdot r_2 \cdot [K_b + (1 - K_b) \cdot \left(\frac{1}{r_2}\right)^u]$$

3.3.3.2 Calculation

Generally, $F_{e,x}$ and $F_{e,y}$, the enlarging factors in the x and y directions, are different. If we magnify a node by $F_{e,x}$ and $F_{e,y}$ in the x and y directions respectively, the aspect ratios of nodes will, in general, be changed after a basic-operation. Such changes could

confuse the user. To maintain the aspect ratio, zoomed nodes must be scaled up by the same factor in both x and y directions. Therefore we introduce a parameter, $F_{e,xy}$, called **common enlarging factor** for zoomed nodes. We calculate $F_{e,x}$ and $F_{e,y}$ by using Equation (3.9) respectively, and set $F_{e,xy}$ to the minimum value of $F_{e,x}$ and $F_{e,y}$ to guarantee that all zoomed nodes are within the Parent-node after a basic-operation. Part C of Table 3.1 shows an example calculating $F_{e,x}$, $F_{e,y}$, and $F_{e,xy}$.

3.3.4 Shrinking Factor F_s

To use screen space most efficiently, the sum of enlarging segments and shrinking segments should equal the length of the Parent-node, after the basic-operation (refer to Figure 3.3). Therefore,

$$F_{e,xy} \cdot S_z + F_s \cdot (L_p - S_z) \equiv L_p \quad (3.11)$$

From this requirement, we derive the formula for F_s as:

$$F_s = \frac{1 - F_{e,xy} \cdot r_3}{1 - r_3} \quad (3.12)$$

where r_3 has been defined in Equation (3.3).

As with the enlarging factor F_e , in general, $F_{s,x}$ and $F_{s,y}$ calculated from Equation (3.12) are different. To maintain the aspect ratio, un-zoomed nodes must be scaled down by the same factor in both x and y directions. Therefore we introduce a parameter, $F_{s,xy}$, called the **common shrinking factor** for un-zoomed nodes. The

calculation of $F_{s,xy}$ is more complex than $F_{e,xy}$. We first calculate $F_{s,x}$ and $F_{s,y}$ using Equation (3.12). Then $F_{s,xy}$ is calculated by the following algorithm to guarantee that, after a basic-operation, all un-zoomed nodes: (a) are within the Parent-node; and (b) do not shrink to a point.

ALGORITHM for $F_{s,xy}$:

```

IF       $minimum(F_{s,x}, F_{s,y}) > 0$ 
THEN    $F_{s,xy} = minimum(F_{s,x}, F_{s,y});$ 
IF       $(minimum(F_{s,x}, F_{s,y}) = 0) \text{ AND } (maximum(F_{s,x}, F_{s,y}) > 0)$ 
THEN    $F_{s,xy} = maximum(F_{s,x}, F_{s,y});$ 
IF       $maximum(F_{s,x}, F_{s,y}) = 0$ 
THEN    $F_{s,xy} = 0.2;$ 

```

This algorithm is developed in Appendix A, and Part D of Table 3.1 shows an example calculating $F_{s,x}$, $F_{s,y}$, and $F_{s,xy}$.

We must realize the following important difference between the enlarging factor group (including $F_{e,x}$, $F_{e,y}$, and $F_{e,xy}$), and the shrinking factor group (including $F_{s,x}$, $F_{s,y}$, and $F_{s,xy}$). In the enlarging factor group, $F_{e,x}$ and $F_{e,y}$ could be “forgotten” after we obtain $F_{e,xy}$, and only $F_{e,xy}$ is used in the following algorithms, such as those for transforming the nodes’ sizes and positions in a basic-operation. However, $F_{s,x}$ and $F_{s,y}$ still take an important role in the following algorithms, even after we obtain $F_{s,xy}$. $F_{s,xy}$ is used for the transformation of nodes’ sizes, while $F_{s,x}$ and $F_{s,y}$ are used for the transformation of nodes’ positions. This is because only the pair of $F_{s,x}$ and $F_{e,xy}$ satisfies the “using screen space efficiently” requirement (see Equation (3.11)) in the x direction, and only the pair of $F_{s,y}$ and $F_{e,xy}$ does so in the y direction. In general,

the pair $F_{s,xy}$ and $F_{e,xy}$ cannot fit in Equation (3.11) in both x and y directions.

3.3.5 Balance Factor K_b

In Equation (3.8) and Equation (3.9), a balance factor K_b provides a control parameter. We explain its geometric meaning with Figure 3.7, which shows three R_z vs r_4 lines with $K_b = 0.2, 0.5, 0.8$ respectively.

1. K_b determines the threshold value mentioned in Equation (3.6), because $R_z \rightarrow K_b$, when $r_4 \rightarrow 0$ in Equation (3.8). As illustrated in Figure 3.7, K_b is the intersection of the R_z vs r_4 line and the R_z axis, and $0 < K_b < 1$.
2. K_b controls the ratio of the area of the zoomed Current-nodes to that of the Parent-node (and in turn, the ratio of detail to context). The bigger K_b is, the larger the detail area. From a one-dimensional point of view, this ratio is simply R_z . If we re-arrange Equation (3.8) as

$$R_z = r_4 + (1 - r_4) \cdot K_b \quad (3.13)$$

We see that when the number of zoomed nodes is fixed (i.e. r_4 is fixed), R_z is proportional to K_b . For example, in Figure 3.7, when $r_4 = 0.2$, R_z varies from 0.36 to 0.84 as K_b increases from 0.2 to 0.8.

K_b is a useful user-controllable parameter, allowing adjustment of the relative emphasis on detail and context, which depends on the circumstance of the problem. Figure 3.8 shows the comparison of layouts with K_b increasing from 0.2 to 0.8.

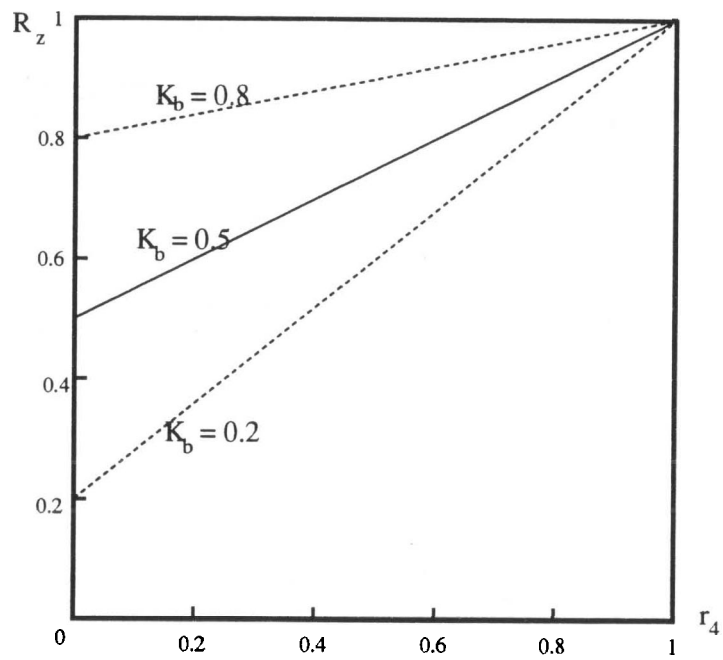


Figure 3.7: $R_z = K_b + (1 - K_b) \cdot r_4$ with $K_b = 0.2, 0.5, 0.8$ respectively.

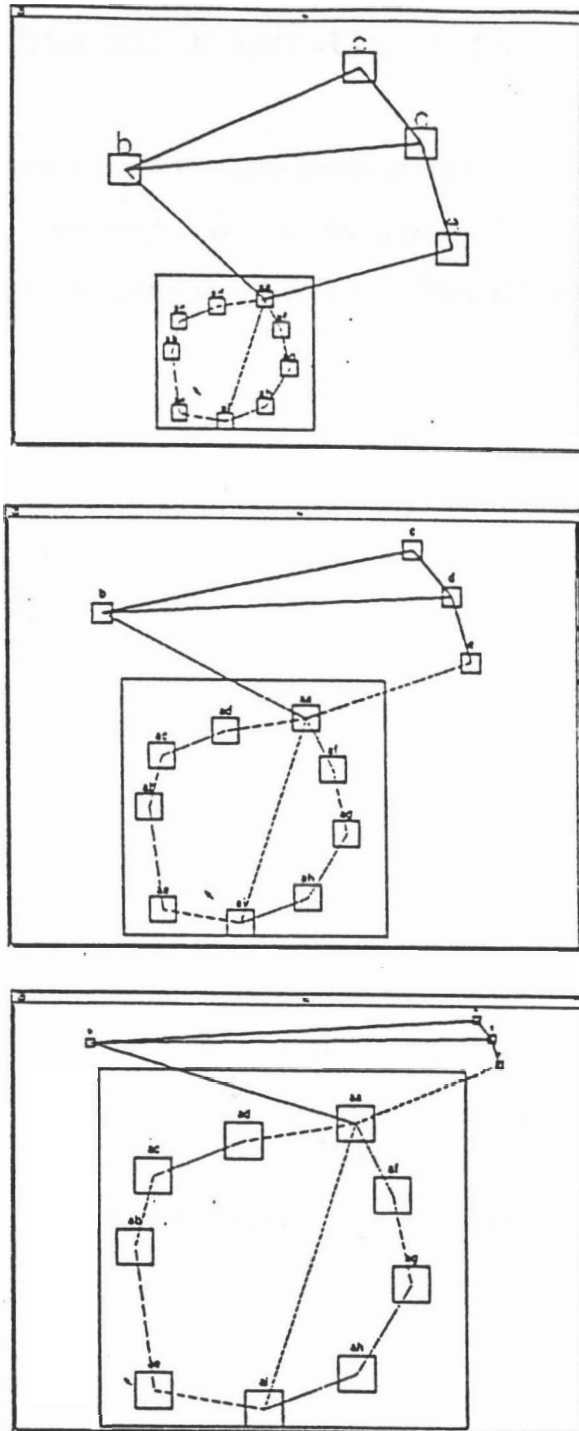


Figure 3.8: Comparison of layouts with different K_b .
 (From top to bottom, $K_b = 0.2, 0.5, 0.8$ respectively.) .

3.4 Algorithm for Basic-Operation

Though a basic-operation involves three-levels of nodes and two levels of networks (see Subsection 3.2.2), we need only consider how to transform the Current-nodes from the normal (uniform) layout to a VZ layout. This is because:

- the Parent-node is not changed by the basic-operation;
- the Sub-nodes, the children of Current-nodes to be zoomed, will all be (uniformly) unzoomed;
- links in the Current-network and Sub-networks are defined simply as straight lines joining the node's centers (see Subsection 3.2.1).

Transformation of the Current-nodes includes changing both sizes and positions of these nodes.

3.4.1 Transforming Node's Size

After a basic-operation, zoomed nodes are magnified by F_e , and un-zoomed nodes are demagnified by F_s . To maintain the aspect ratio of a node, $F_{e,xy}$ and $F_{s,xy}$ are used here.

ALGORITHM transforming node's size:

1. For zoomed nodes:

$$L'_x = F_{e,xy} \cdot L_x \quad (x \text{ direction}) \quad (3.14)$$

$$L'_y = F_{e,xy} \cdot L_y \quad (y \text{ direction}) \quad (3.15)$$

2. For un-zoomed nodes:

$$L'_x = F_{s,xy} \cdot L_x \quad (x \text{ direction}) \quad (3.16)$$

$$L'_y = F_{s,xy} \cdot L_y \quad (y \text{ direction}) \quad (3.17)$$

where L and L' are the length of a node before and after a basic-operation respectively.

3.4.2 Transforming Node's Position

As described in Subsection 3.1.2 and Figure 3.3, we divide the Parent-node length into two types of segments, enlarging and shrinking, using a 1D view. To transform all Current-node positions, we first determine the enlarging and shrinking segments to build a sorted list, and then calculate the center position of a node using this list.

ALGORITHM for determining enlarging and shrinking segments and building a sorted list:¹

1. Project all nodes to be zoomed onto the x-axis to form a list: $list_{z,x}$.
2. CHECK-MERGE segments in $list_{z,x}$ to obtain $list'_{z,x}$ (see CHECK-MERGE in Subsection 3.3.1.1) to guarantee there are no overlaps between the enlarging segments in $list'_{z,x}$.
3. Sort the segments in $list'_{z,x}$ from left to right.
4. Shrinking segments are the spaces either between enlarging segments or between an enlarging segment and a boundary of the Parent-node.
 - (a) The left boundary of the first shrinking segment is the left boundary of the Parent-node, and the right boundary is the left boundary of the first enlarging segment in $list'_{z,x}$.
 - (b) The left boundary of the i^{th} shrinking segment is the right boundary of the $(i - 1)^{th}$ enlarging segment in $list'_{z,x}$, and the right boundary is the left boundary of the i^{th} enlarging segment, here

$$2 \leq i \leq (\text{the_number_of_enlarging_segments})$$
 - (c) The left boundary of the last shrinking segment is the right boundary of the last enlarging segment, and the right boundary is the right boundary of the Parent-node.
5. Build a list including all enlarging and shrinking segments, and sort these segments from left to right. This is a **sorted all-segment list**.

¹The first two steps of the algorithm are the same as the ALGORITHM for S_z described in Subsection 3.3.1.1.

ALGORITHM transforming node's position:

Let x_i and x'_i be the center positions of a node (zoomed or un-zoomed) before and after a basic-operation.

1. Initialize $x'_i = \text{left_boundary_of_parent_node}$.
2. In the sorted all-segment list, for each segment left of x_i :
 - if it is an enlarging segment

$$x'_i = x'_i + F_{e,xy} \cdot (\text{length_of_the_segment});$$
 - else $x'_i = x'_i + F_{s,x} \cdot (\text{length_of_the_segment});$

3. For the segment containing x_i :

If it is an enlarging segment

$$x'_i = x'_i + F_{e,xy} \cdot (\text{dist_from_left_boundary_of_the_seg_to_}x_i);$$

$$\text{else } x'_i = x'_i + F_{s,x} \cdot (\text{dist_from_left_boundary_of_the_seg_to_}x_i);$$

3.4.3 An Example

Figure 3.9 is an example illustrating the algorithm for a basic-operation. Figure 3.9(a) is a normal (uniform) layout before a basic-operation, the same as that in Figure 3.5. Figure 3.9(b) is the variable zoom layout after the basic-operation. Table 3.2 illustrates the transformation of nodes' sizes. Table 3.3 determines enlarging and shrinking segments and builds the sorted all-segment list. Table 3.4 shows transforming the position of node 'c' as a demonstration.

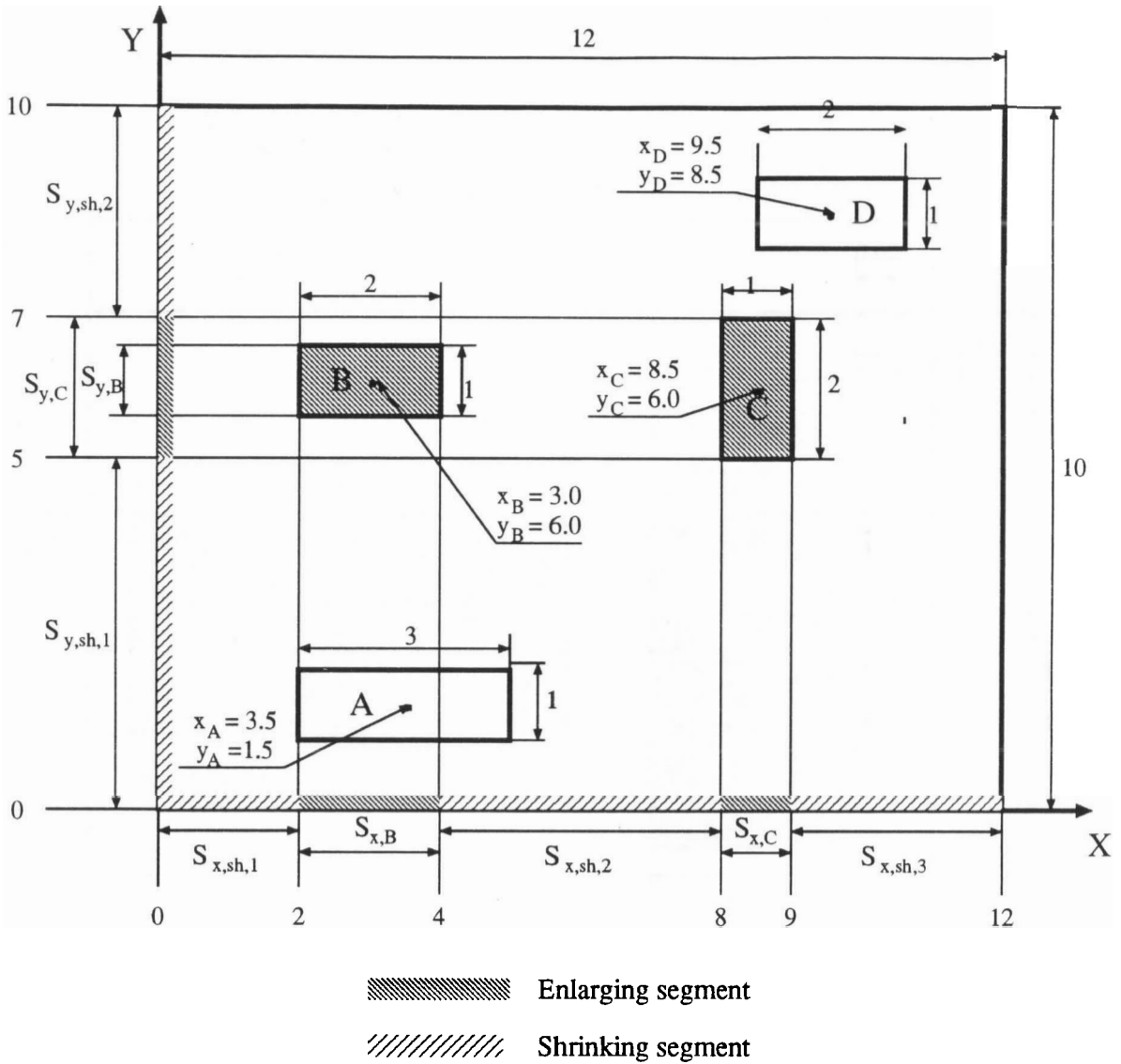


Figure 3.9: (a) A normal layout before a basic-operation. (Shaded rectangles are the nodes to be zoomed. Variables x_A and y_A are the center coordinates of node 'A', and similar variables are used for node 'B' to 'D'. In the x-axis, $S_{x,B}$ and $S_{x,C}$ are enlarging segments; $S_{x,sh,1}$ to $S_{x,sh,3}$ are shrinking segments; and the numeric values below are boundary values of these segments. In the y-axis, all the symbols and numbers have similar meanings.)

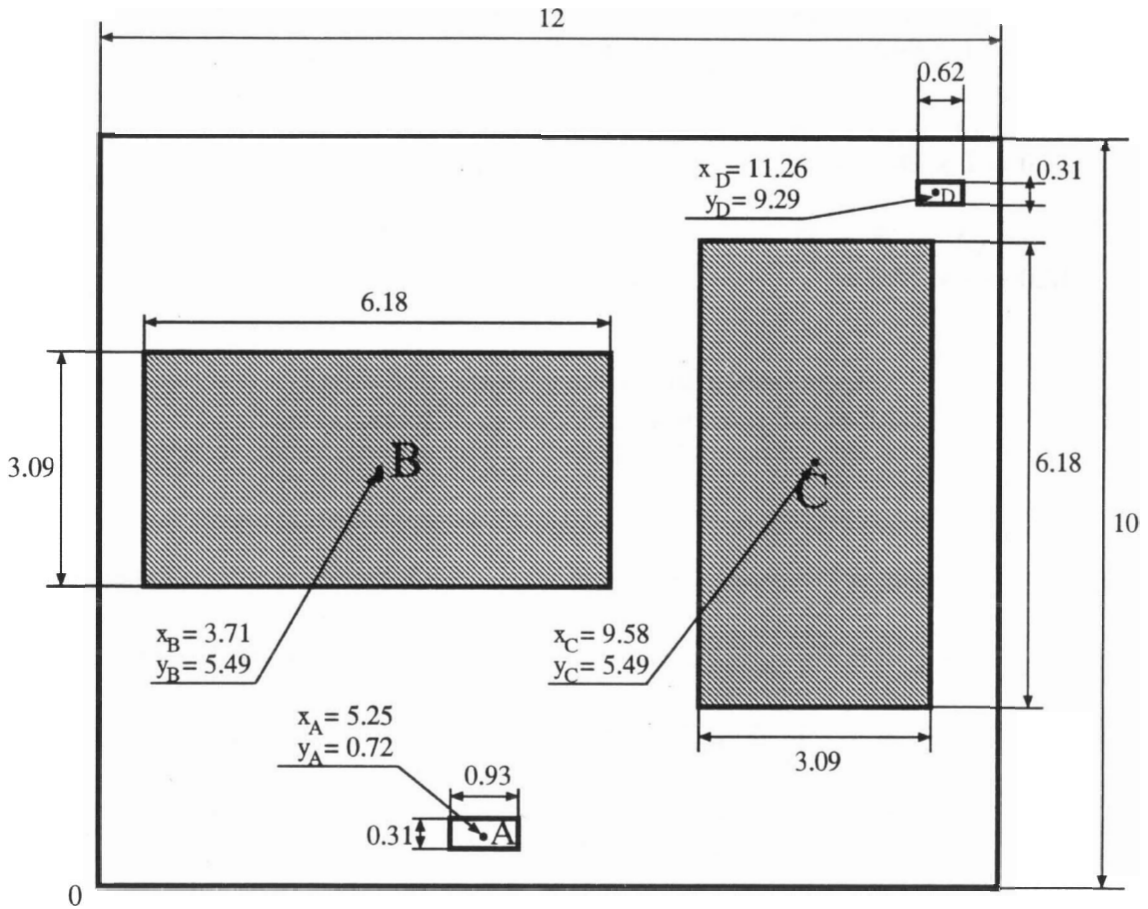


Figure 3.9: (b) A variable zoom layout after the basic-operation.
 (Shaded squares are the zoomed nodes.)

Node	Status	x-direction	y-direction
'A'	un-zoomed	$L'_x = F_{s,xy} \cdot L_x$ $= 0.31 \times 3 = 0.93;$	$L'_y = F_{s,xy} \cdot L_y$ $= 0.31 \times 1 = 0.31;$
'B'	zoomed	$L'_x = F_{e,xy} \cdot L_x$ $= 3.09 \times 2 = 6.18;$	$L'_y = F_{e,xy} \cdot L_y$ $= 3.09 \times 1 = 3.09;$
'C'	zoomed	$L'_x = F_{e,xy} \cdot L_x$ $= 3.09 \times 1 = 3.09;$	$L'_y = F_{e,xy} \cdot L_y$ $= 3.09 \times 1 = 6.18;$
'D'	un-zoomed	$L'_x = F_{s,xy} \cdot L_x$ $= 0.31 \times 2 = 0.62;$	$L'_y = F_{s,xy} \cdot L_y$ $= 0.31 \times 1 = 0.31;$

Table 3.2: Transformation of nodes' sizes.
(The values of $F_{e,xy}$ and $F_{s,xy}$ are from Table 3.1.)

Procedure	x-direction	y-direction
1. Project nodes to be zoomed:	$list_{z,x} = [S_{x,B}, S_{x,C}];$	$list_{z,y} = [S_{y,B}, S_{y,C}];$
2.CHECK-MERGE $list_z$:	$list'_{z,x} = [S_{x,B}, S_{x,C}];$	$list'_{z,y} = [S_{y,C}];$
3. Sort $list'_z$:	$list_{z,x} = [S_{x,B}, S_{x,C}];$	$list_{z,y} = [S_{y,C}];$
4. Determine shrinking segments:	$S_{x,sh,1}, S_{x,sh,2}, S_{x,sh,3};$	$S_{y,sh,1}, S_{y,sh,2};$
5. Build sorted all-segment list:	$[S_{x,sh,1}, S_{x,B}, S_{x,sh,2},$ $S_{x,C}, S_{x,sh,3}];$	$[S_{y,sh,1}, S_{y,C}, S_{x,sh,2}];$

Table 3.3: Determining enlarging and shrinking segments and building the sorted all-segment list.

(All segments are marked in Figure 3.9(a).)

Initialization/Segments	Calculation:
(Calculating x'_C)	
Initialization:	$x'_C = \text{left_bdy_of_parent_node} = 0;$
$S_{x,sh,1}$ (left to x_C , shr-seg.)	$x'_C = x'_C + F_{s,x} \cdot L_{x,sh,1} = 0 + 0.31 \times 2 = 0.62$
$S_{x,B}$ (left to x_C , enl-seg.)	$x'_C = x'_C + F_{e,xy} \cdot L_b = 0.62 + 3.09 \times 2 = 6.80$
$S_{x,sh,2}$ (left to x_C , shr-seg.)	$x'_C = x'_C + F_{s,x} \cdot L_{x,sh,1} = 6.80 + 0.31 \times 4 = 8.04$
$S_{x,C}$ (containing x_C , enl-seg.)	$x'_C = x'_C + F_{e,xy} \cdot (\text{dist_from_left_bdy_to_}x_C)$ $= 8.04 + 3.09 \times 0.5 = 9.58$
(Calculating y'_C)	
Initialization:	$y'_C = \text{bottom_bdy_of_parent_node} = 0;$
$S_{y,sh,1}$ (below to y_C , shr-seg.)	$y'_C = y'_C + F_{s,y} \cdot L_{y,sh,1} = 0 + 0.48 \times 5 = 2.40$
$S_{y,C}$ (containing y_C , enl-seg.)	$y'_C = y'_C + F_{e,xy} \cdot (\text{dist_from_botom_bdy_to_}y_C)$ $= 2.40 + 3.09 \times 1 = 5.49$

Table 3.4: Transformation of a node's position.

(The position of node 'C' in Figure 3.9(a) is transformed here, as an example of the transformation algorithm. The values of $F_{e,xy}$, $F_{s,x}$, and $F_{s,y}$ are from Table 3.1.)

3.5 Varying Balance Factor with Level

The variable zoom method often produces multiple levels of context. Should the balance factor K_b be a constant for all levels? If not, how should K_b vary with level?

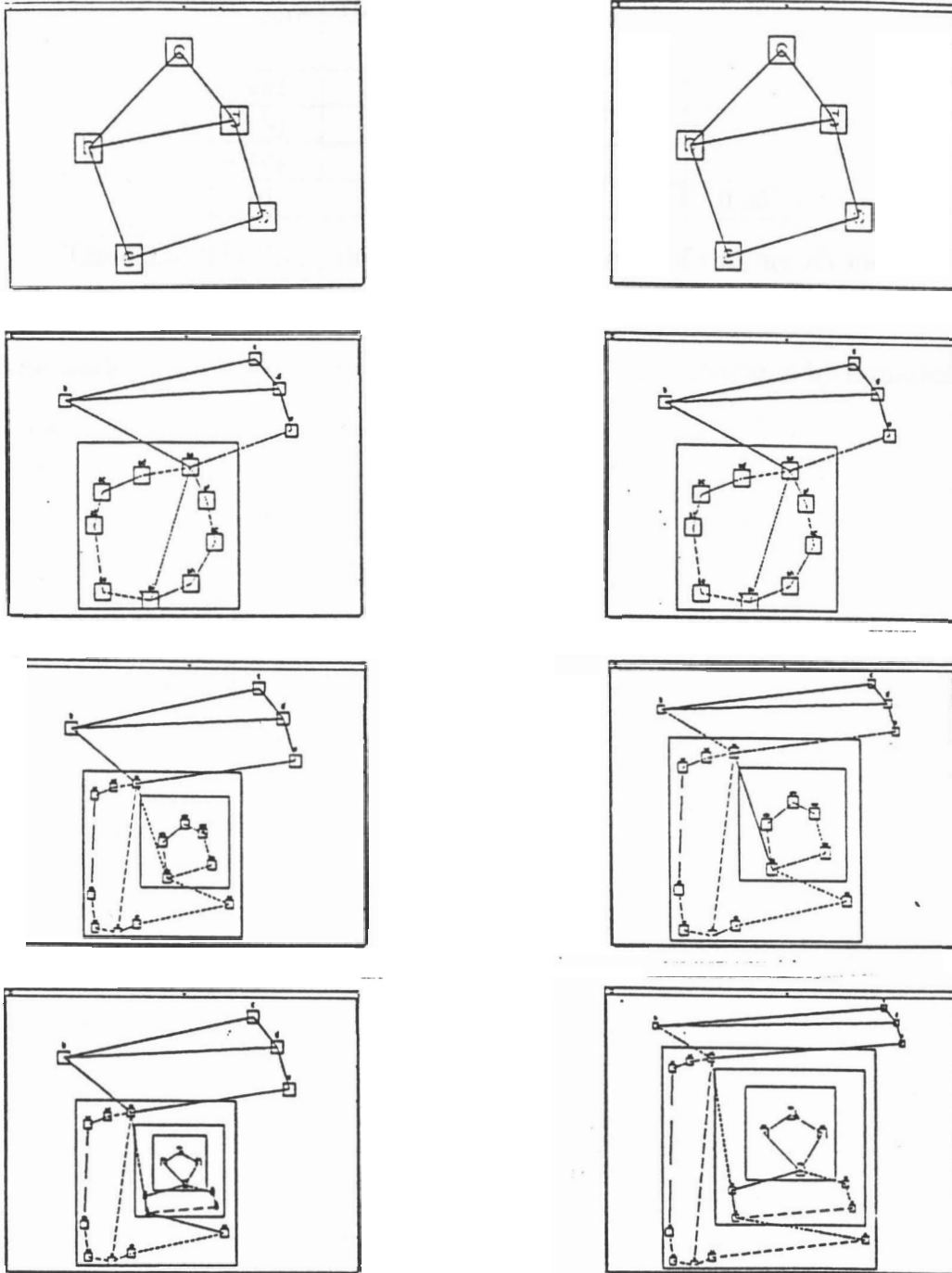
Figure 3.10(a) (the left column) illustrates an example using a constant balance factor. As we move the detailed view from the second level to the fourth, the “importance” of the top network decreases, and we might expect the space it occupies to decrease accordingly. However, a constant K_b maintains the size of the top network despite the increasing depth of the detailed view. To improve the context display, we use a varying K_b to reduce the space occupied by a context level with its distance from the detailed level.

The detailed view is assumed at the n^{th} level, and $K_{b,i}$ is the balance factor at the i^{th} level ($i = 1 \dots n-1$). We suggest:

$$K_{b,i} = \frac{(n-i)}{(n-i)+1} \quad (i = 1 \dots n-1) \quad (3.18)$$

Here, $(n-i)$ is the distance from the i^{th} level of context to the n^{th} level of detail. $K_{b,i}$ increases with the distance, in other words, the size of a level of context decreases with its distance from the detail. Equation (3.18) also keeps $K_{b,i}$ greater than or equal to 0.5, and less than 1. The derivation of Equation (3.18) is described in Appendix B in detail.

Figure 3.10(b) (the right column) illustrates an example of the varying K_b method. When the detailed view moves from the second level to the fourth, the context of the



(a) Constant K_b ($K_b = 0.5$)

(b) Variable K_b (see Table 3.5).

Figure 3.10: Compare constant K_b with variable K_b .

(From top to bottom, the detailed view moves from the first level to the fourth level)

Figure	Detailed level n	Balance factors $K_{b,i}$		
		$K_{b,1}$	$K_{b,2}$	$K_{b,3}$
(a)	1			
(b)	2	0.50		
(c)	3	0.67	0.50	
(d)	4	0.75	0.67	0.50

Table 3.5: The $K_{b,i}$ values used in the example of varying K_b method.

top network is reasonably reduced. The values of K_b are calculated by Equation (3.18) and shown in Table 3.5.

It is important to realize that K_b is a user-controllable parameter. The value of K_b at each level essentially depends on user requirements and network configuration, and should be determined through human factors experiments and users' experiences. This section only describes a principle, gradually demagnifying levels of context with the distance from the detailed view, and provides default values of K_b using Equation (3.18).

CHAPTER 4

A PRELIMINARY EXPERIMENT

In order to test the value of the variable zoom technique, a preliminary human factors experiment compared the performance of the variable zoom (VZ) method vs a standard full screen zoom (FZ) method. The software package used in this experiment was provided by the Computer Graphic Research Laboratory of Simon Fraser University. The main part of the package, representing an implementation of the algorithm described in this thesis, was developed by the author. The experiment was performed by D. Schaffer, S. Dubs and M. Roseman in the University of Calgary and is described in detail in [Schaffer 92] [IGI 92-0061]. Only an overview of the experiment is given here together with a summary of the results which showed that the VZ method was significantly better than the standard FZ method in this experiment.

4.1 Description of the Experiment

Subjects were given a hierarchical (pseudo) telephone network and asked to:

1. Identify a “bad” link in a sequence of links connecting two telephones; and
2. Reroute the call, i.e. reconnect the two telephones by using other links.

4.1.1 Zoom Methods

Subjects performed these tasks using either the VZ method described in Chapter 3, or a standard FZ method developed by the author for this experiment. In the latter, a node selected by the subject was magnified to essentially full screen size (hence, its “full screen zoom” name) without context. Figure 4.1 and Figure 4.2 show the examples of the FZ and VZ views in the experiment.

The network used was a four level hierarchical network and consisted of about 200 nodes and 200 links. “Logical” nodes and “physical” nodes (refer to Subsection 3.1.1) were visually distinguishable, and only logical nodes could be zoomed/unzoomed. Logical and physical links were also distinguished, while, to prevent ambiguities, only physical links could be selected/deselected. Different colors were used to distinguish the links’ status, green for selected and black for deselected.

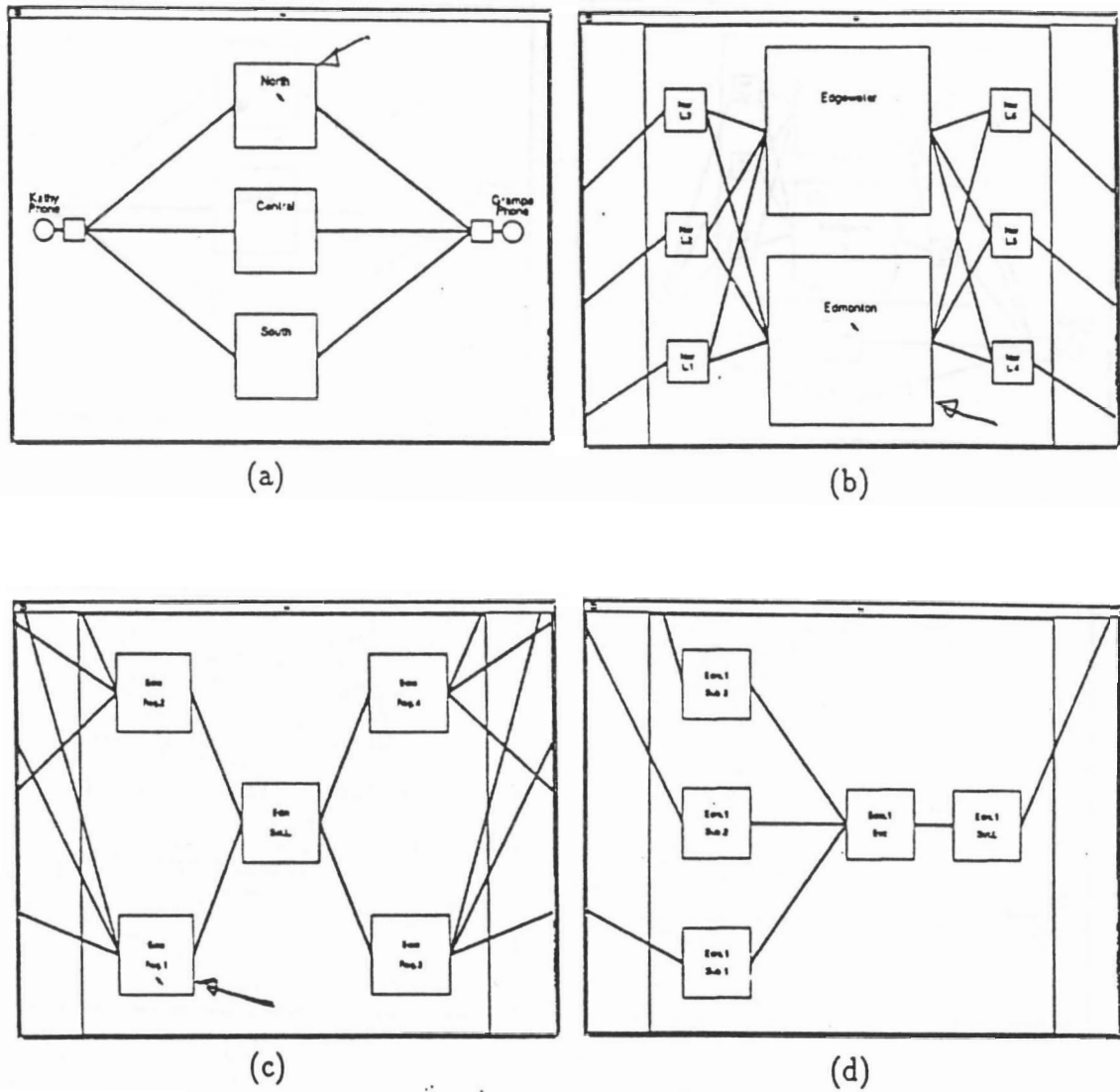


Figure 4.1: Examples of the FZ views in the human factors experiment.

- (a) Top network;
- (b) 'North' node from (a) zoomed;
- (c) 'Edmonton' node from (b) zoomed;
- (d) 'Edm.Reg.1' node from (c) zoomed.

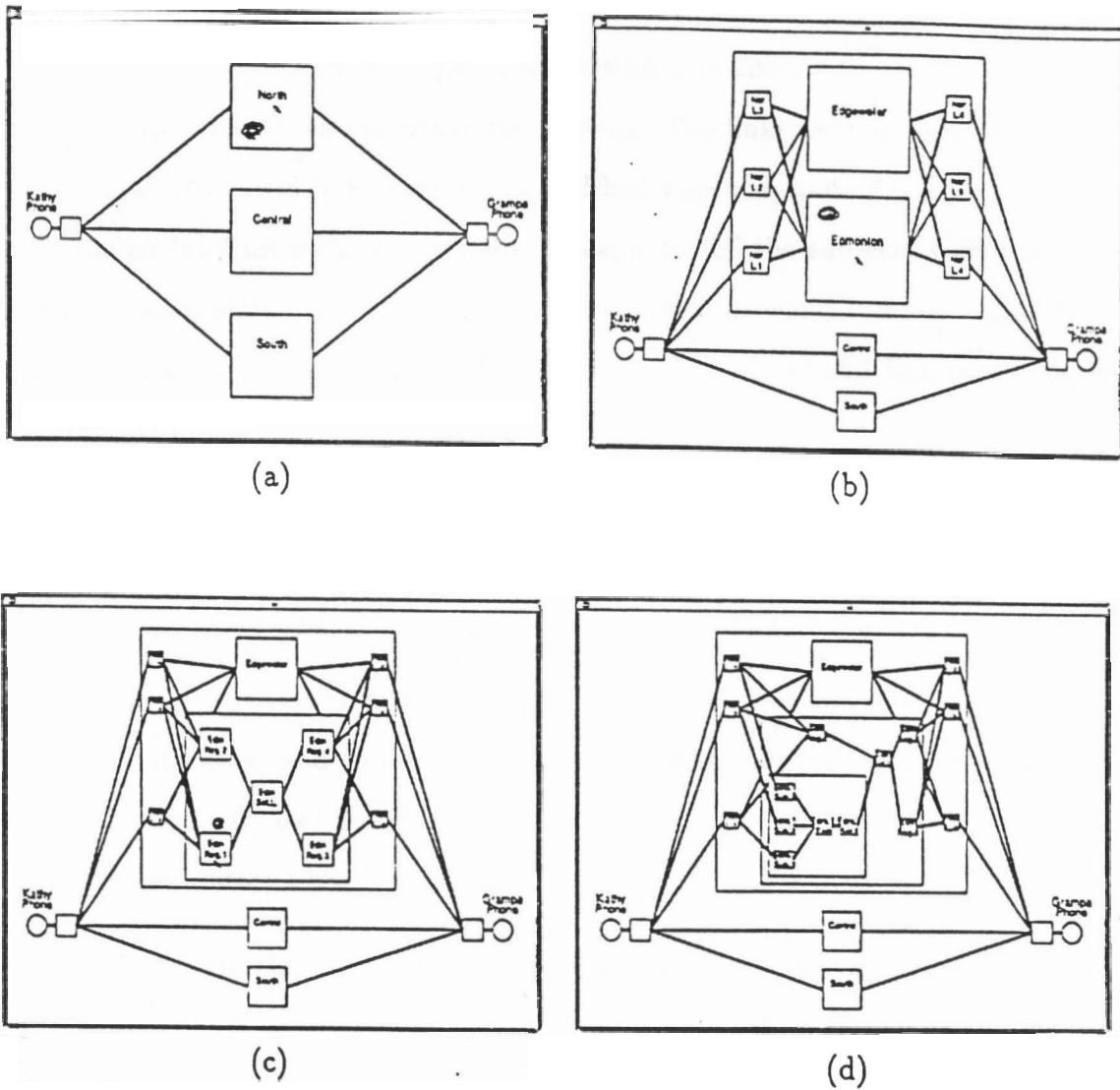


Figure 4.2: Examples of the VZ views in the human factors experiment.

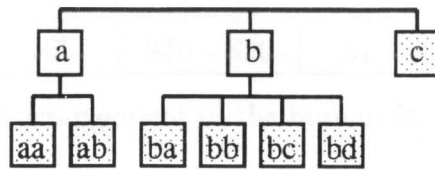
- (a) Top network;
- (b) 'North' node from (a) zoomed;
- (c) 'Edmonton' node from (b) zoomed;
- (d) 'Edm.Reg.1' node from (c) zoomed.

4.1.2 Task

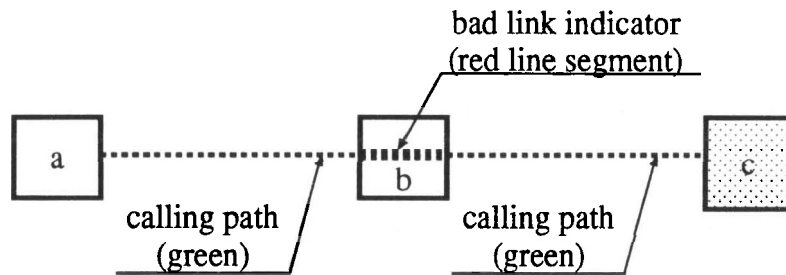
Each task performed by a subject started with a telephone call between Kathy and her Grampa, the two end points of the network. The links used by the call were shown in green or red: good links were green, and bad were red; unused links were black. At the higher level network, we used an indicator to tell the subjects where they could find the undisplayed low level bad link. The indicator was a red line segment in the middle of all higher level nodes containing the lower level bad link (see node 'b' in Figure 4.3(b)).

Subjects, acting as telephone network operators, were asked to find and solve a problem with an ongoing call.

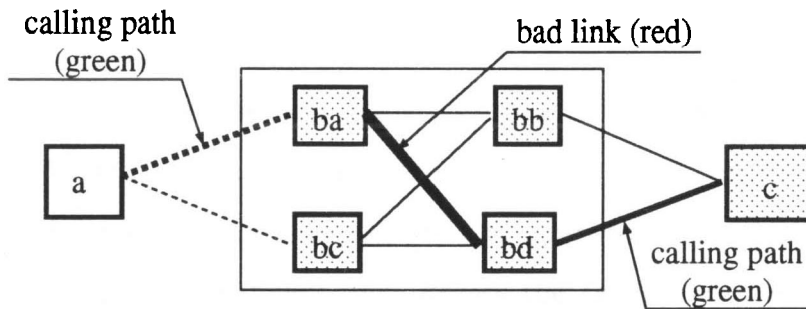
1. They were asked to find the (single, lowest level) bad link by navigating through the network, and to identify it by selecting it with a mouse. After the identification, the bad link turned from red to yellow and the subjects selected the screen button "break found".
2. They were then asked to build a new route by selecting and deselecting appropriate links. Selecting turned a link from black to green; deselecting turned green to black. The new route was completed when a path of green links directly connected Kathy to her Grampa. Then the subjects selected the screen button "reroute complete" to complete the task.



(a) node tree of this example.



(b) the higher level network



(c) after node 'b' be zoomed



Figure 4.3: An example of bad link indicator.
 (The colors of line on the computer screen are expressed by the thicknesses of lines here. The thickest lines are red lines, the medium ones are green, and the thinnest are black. Dashed lines represent logical links and solid lines represent physical ones.) .

Method	Problem A	Problem B
VZ	S1 - S9	S10 - S20
FZ	S10 - S20	S1 - S9

Table 4.1: The arrangement of the human factors experiment.

4.1.3 Subjects and Experiment Method

20 subjects were selected from senior undergraduate students, graduate students, or faculty in computer science. They were familiar with graphical user interfaces and general data structures. Each was asked to solve two problems (A and B). They were divided into two groups, one included 9 subjects, the other 11 subjects. The first group solved problem A with the FZ method first and problem B with the VZ method second; the second group did the reverse. Problem A was the first in all cases, as shown in Table 4.1.

The data, collected by the data logging software¹, was analyzed by using a 2×2 factor ANOVA design, with the two factors: zoom method and the order of zoom method [Schaffer 92]. In addition, qualitative comparisons between the two methods were obtained from subjects' comments during the experiment and from questionnaires after each problem was completed.

¹This software was developed by L. Bartram, and T. Brooks.

4.2 Results

The means and standard deviations of performance time are shown in Figure 4.4(a), with a plot of the means. The analysis of variance (ANOVA) released that the method used (VZ or FZ) was a significant factor affecting running time ($F[1,18] = 9.91$, $p < 0.006$) [Schaffer 92]. We observe that **the average performance time used in the VZ method is only 63% of that used in the FZ method.**

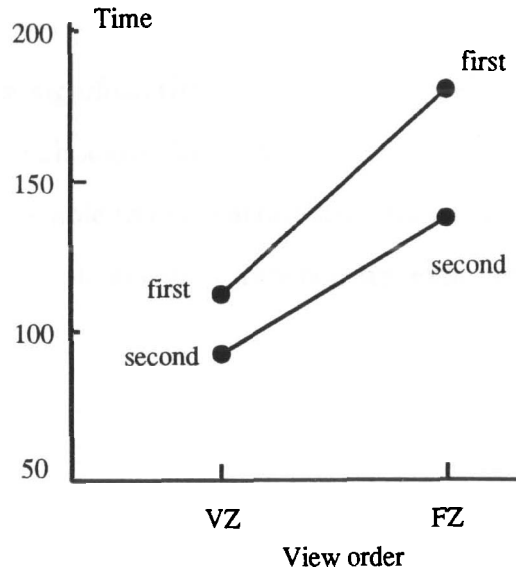
The number of navigation operations required, measured by the number of “zoom” actions performed by the subjects to solve the problems, was shown in Figure 4.4(b). ANOVA released that the number of zooms was also significantly affected by the method used ($F[1,18] = 18.29$, $p < 0.001$) [Schaffer 92]. We observe that **the number of zooms used in the VZ is only 58% of that used in the FZ.**

Qualitative feedback from the subjects showed that 80% greatly preferred the VZ method to the FZ method. Typical comment on the VZ method were:

- “easier to work with, didn’t get lost, context available”
- “visualization of whole system, problem with needed details, use split screen both active, use both at same time”
- “don’t have to remember”
- ...

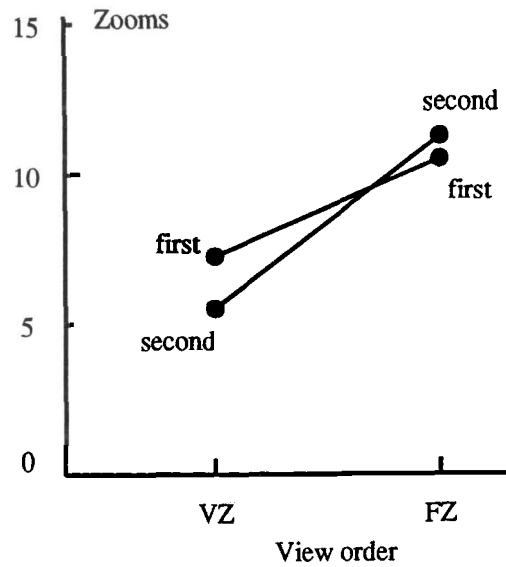
View	Order	Mean	Std dev
VZ	----	101.9	59.5
FZ	----	161.2	71.6
VZ	First	113.5	72.9
FZ	Second	136.6	62.9
FZ	First	181.3	74.7
VZ	Second	92.4	47.6

(unit: second)



(a) Performance time

View	Order	Mean	Std dev
VZ	----	6.3	2.7
FZ	----	10.9	4.3
VZ	First	7.2	3.1
FZ	Second	11.4	5.8
FZ	First	10.5	3.1
VZ	Second	5.5	2.2



(a) Number of zooms

Figure 4.4: Human factors experiment results [Schaffer 92][IGI 92-0061].

Based on this human factors experiment, the conclusion was that:

The variable zoom method was significantly better than the full screen zoom method for navigating through a graphical structure. Using the variable zoom method, “subjects were able to concentrate directly on the task itself, resulting in quicker navigation and less unnecessary exploration” [Schaffer 92]

CHAPTER 5

DISCUSSION

5.1 Comparisons with Other Fisheye View Methods

This section compares the VZ method with other fisheye view methods described in Chapter 2 (refer to Figure 2.1), in the order of fisheye lens transformation, DOI approach, and Perspective Wall.

The example used in the fisheye lens transformation [Sarkar 92] displayed a large network with an impressive image. However, the VZ method has two main advantages over it:

1. With one or more focused nodes, [Sarkar 92] displayed all the other nodes of the network as a context, unless a particular node's "display" value fell below a

threshold. This can result in an excessively crowded context (refer to Figure 2.4), and make the method difficult for very large networks (e.g. over 1,000 nodes). The VZ technique uses higher level nodes (icons) as a context, omitting the lower level detail not of current interest. Using this hierarchical abstraction [Fairchild 88] [Card 91], the VZ method greatly simplifies the context display and increases the richness of the represented information space.

2. Sarkar *et al* also distorted the small area around a focus, when distorting the whole network, which could confuse the user. The VZ method scales the focus region uniformly. It should be easier for the user to understand, recalling that the traditional normal (uniform) layout works very well for a **small** information space¹.

It is interesting to compare the hierarchical node trees, illustrated in Figure 3.2, with the tree structure used in Furnas' DOI approach, described in Figure 2.3. They are very similar, each node in Figure 3.2 corresponds an object in Figure 2.3.

From Figure 5.1(a) to (c), for all nodes in a hierarchical network, we calculate the distances from the focused node 'ca', a *priori* importance, and the DOI values respectively, in the same method as did [Furnas 86] (see Subsection 2.3.1 and Figure 2.3). Comparing Figure 5.1(c) and (d), we observe that:

1. the most interesting nodes, with the highest DOI value (-2), are exactly the
-

¹Here, "small information space" means that its whole image can be perceived as a unit and displayed on a single window with sufficient resolution.

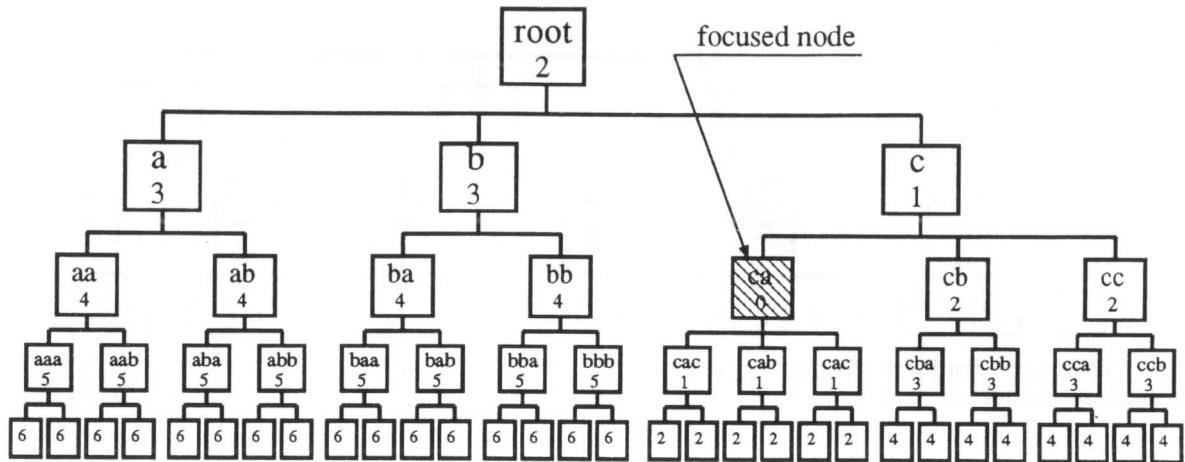
zoomed nodes in the VZ method, and their immediate sub-networks are displayed;

2. the second most interesting nodes, with the second highest DOI value (-4), are exactly the un-zoomed but visible nodes in the VZ method;
3. the other nodes, with the DOI values lower than (-4), are omitted in the VZ method.

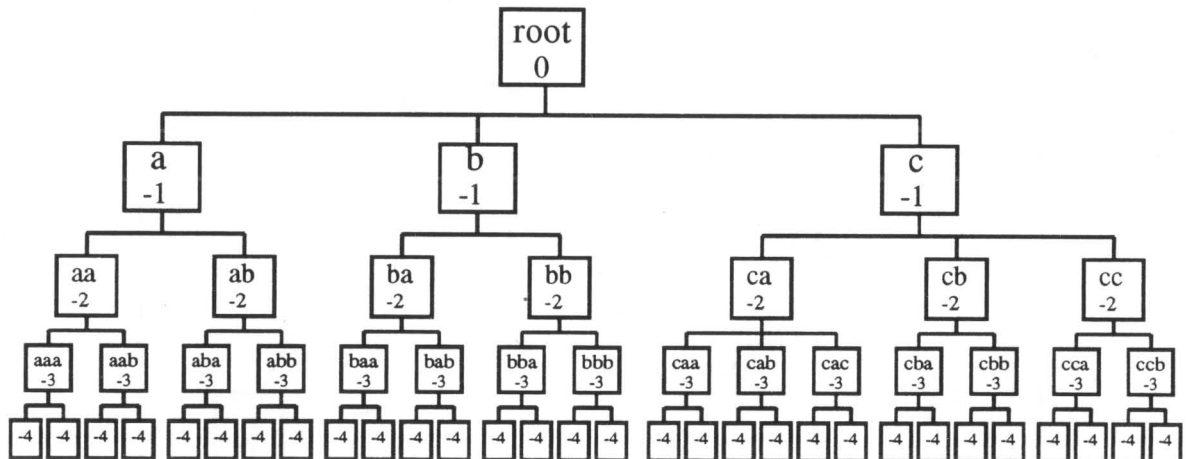
From the DOI point of view, the VZ method automatically (1) sets up the global importance rating, and calculates the distance and the DOI values for each node; (2) sets up a threshold as the second highest DOI value and omits all nodes with the DOI values lower than the threshold; (3) views the nodes with the highest DOI value in detail.

Both the Perspective Wall [Mackinlay 91] and the VZ method display a large information space combining uniformly magnified details with a gradually demagnified context. However, they have several differences:

1. The VZ method uses a hierarchical representation, as described above; the Perspective Wall does not.
2. The Perspective Wall was limited by a linear (1D) structure; the VZ method represents a 2D network. The latter is intrinsically more difficult than the former.
3. The Perspective Wall allowed only one detailed region; the VZ method allows displaying multiple detailed views and their connections. This is especially

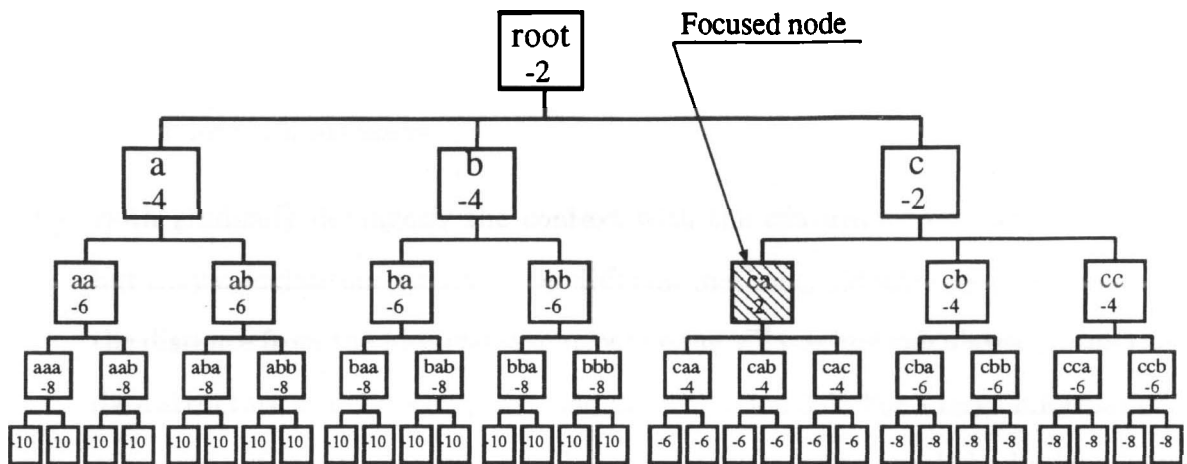


(a) Distance from focused node 'ca'

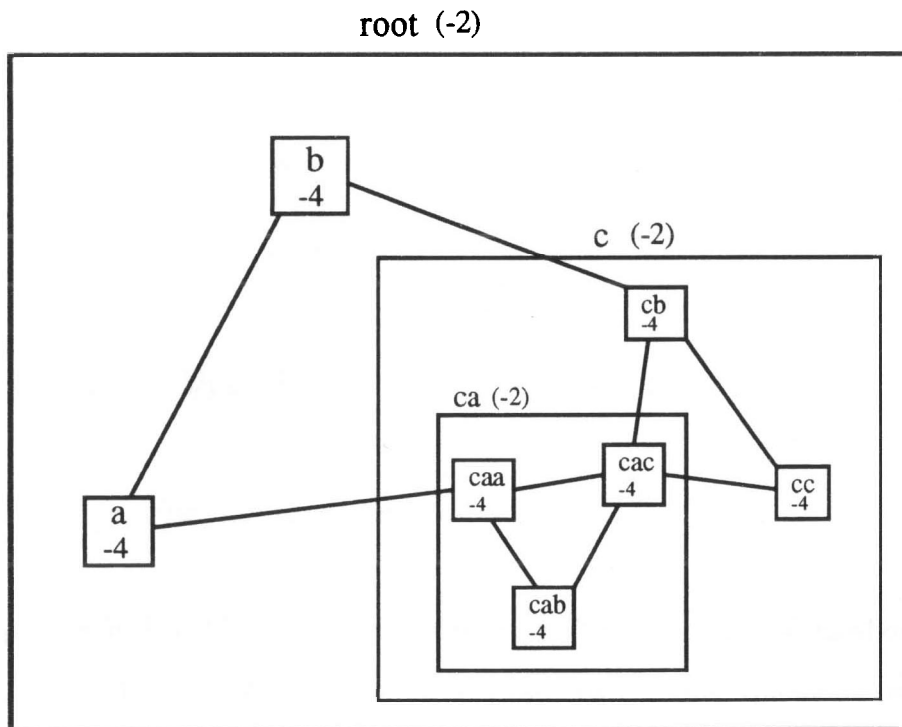


(b) A Priori Importance in the tree

Figure 5.1: The variable zoom method and DOI values. (For a node, the English letter is the name and the digit is the value. The names of the lowest level nodes are ignored.) .



(c) DOI values



(d) the variable zoom view

Figure 5.1: (continued).

important for telecommunication networks and power distribution systems, because operators often need to check or build a path between two or more nodes in different sub-networks.

4. Both gradually demagnify the context with the **distance** from detailed view, but they calculate such distance by different methods. [Mackinlay 91] calculated the distance from the physical layout, while the VZ method calculates it from the data structure as described above and in Subsection 3.5. For a large information space, one of the main differences between the human eye and the brain might be that the eye views the information through physical space, while the brain aggregates the information through an abstract data structure. For example, two geometrically distant pieces of information might be very close in the human mental model. The VZ method seems to fit this model. We speculate that this might help explain the good performance of the VZ method in the human factors experiment described in Chapter 4.

5.2 Future Work

The following are areas of interest for future research.

DOI approach for hierarchical networks: The current VZ method assumes that all nodes and links in the same sub-network have the same importance. All of them are shown when their parent node is zoomed, and omitted when their parent node is un-zoomed. In some cases, these importance values might be different. As mentioned in Subsection 2.3.1, [Mitta 89] calculated the DOI

value of a node by trading off the global importance against the sum of the minimum path distances from all focal points, and only displayed those nodes with DOI values greater than a threshold. We should investigate this kind of DOI approach in the near future and combine it with the VZ method either to improve the VZ view further, or to create a new technique.

Status representation: In hierarchical networks, the status of lower level networks should be reflected in the representation of higher level icons. As illustrated in Figure 4.3, we used a red line segment within higher level nodes (icons) to indicate the lower level bad links in the human factors experiment prototype. In future implementations, we could use a green (or black) line segment as an indicator to present a completed (or incompleted) lower level connection. Other iconic representations, such as alarms, should also be investigated.

Link-based algorithm: In Chapter 3, we used a node-based algorithm for the VZ technique. In some cases, such as in the SLD (Single Line Diagram) of TAU (TransAlta Utilities Corporation) power distribution network, operators sometimes must deal with a link, which is not only a line connecting two nodes but also consists of several real lines. A link-based or a network-based (including node-based and link-based methods) algorithm should be investigated [Meunier 88].

“Manhattan” geometry links: In the current algorithm, a link is simply a straight line between the centers of two nodes. In some cases, such as in the grid map of TAU, links consist of horizontal and vertical segments, called “Manhattan” geometry links. A suitable method for generating such links is desired.

2D derivation: In Chapter 3, we derived the VZ algorithms from a 1D point of view. The results are reasonable. However a 2D derivation should be examined.

General 2D information spaces: The current VZ method is limited to hierarchical network structures. It would be desirable to extend it to general 2D information spaces, such as a geographic map. This is a great challenge for the VZ method.

CHAPTER 6

SUMMARY

Large hierarchical networks are widely used in supervisory control systems, such as those in telecommunications network management, power distribution, and industrial process control. These networks are becoming increasingly difficult for operators to understand and operate correctly. An “information bottleneck” exists between operators and systems.

Traditional pan/zoom method works very well for **small** information spaces, but cause severe navigation problems for **large** information spaces. Several new technologies have been proposed to improve the graphical representation of large 2D information spaces and have met with varying degrees of success. These included the map view, the DOI approach of fisheye views, the fisheye lens transformation, the perspective wall, and the cone tree. After reviewing and evaluating these current technologies, we find that a new appropriate technique is needed to display large hierarchical networks.

Combining hierarchical representations with a generalized fisheye view, this thesis develops a new technique, called a **variable zoom**, to represent large hierarchical networks. Its main features are:

- logically iconic representations of a hierarchical structure.
- integrating and balancing one or more details and one or more levels of context in a single window;
- uniformly magnifying details;
- gradually demagnifying levels of context;
- adjusting magnification/demagnification factors according to the environment and the user's requests;
- allowing multiple detailed views.

We first develop an algorithm for a **basic-operation**, which transform the nodes' sizes and positions in a single level. Three important parameters are derived for the algorithm: the **enlarging factor** and **shrinking factor** are adjusted according to the environment and the user's request; and a **balance factor** is used to balance the ratio of detail and context. When extending the algorithm to multiple levels, we suggest a **varying balance factor** method to gradually demagnify the different levels of context depending on the distance from the detailed view.

Two implementation prototypes have been completed. They were written in C and HOOPS, and worked on both SUN and IRIS workstations. One demonstrated the

variable zoom technique with a four level abstract network. The other was designed for a human factors experiment with a simplified hierarchical telephone network.

A preliminary human factors experiment was performed to compare the variable zoom technique with a traditional uniform zoom technique. The results showed that the variable zoom method was significantly better than a traditional full screen zoom method.

We conclude that the variable zoom technique can improve the display and navigation of large hierarchical networks. It holds significant promise for use in real supervisory control systems. It is currently planned to be implemented into two experimental industrial prototypes in the IGI project, one a power transmission network, the other a telecommunication network [IGI 92-0046] [IGI 92-0087].

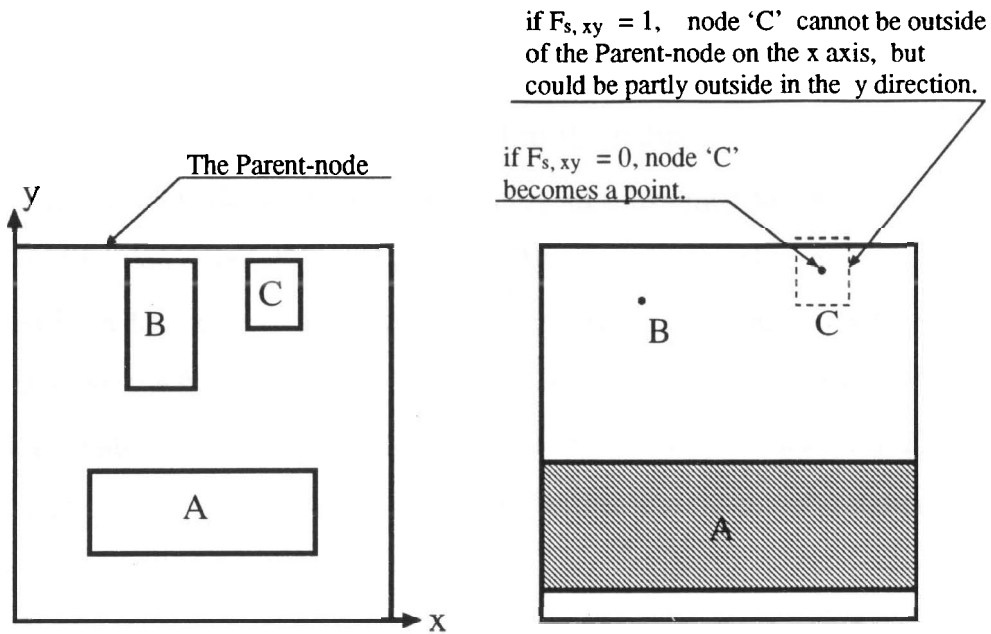
Appendix A

ALGORITHM FOR $F_{s,xy}$

The common shrinking factor $F_{s,xy}$ is the demagnification factor for un-zoomed nodes in both x and y directions, as described in Subsection 3.4.1. The first requirement of $F_{s,xy}$ is that all un-zoomed nodes must be within the Parent-node after a basic-operation, called “within Parent-node” requirement. To satisfy this requirement, a simple method, as that for $F_{e,xy}$ in Subsection 3.3.3.2, is:

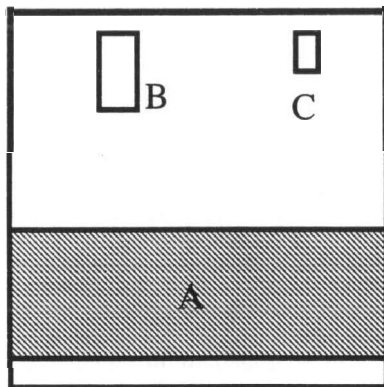
$$F_{s,xy} = \text{minimum}(F_{s,x}, F_{s,y}) \quad (\text{A.1})$$

However, different from $F_{e,x}$ and $F_{e,y}$ (they are always greater than 1), $F_{s,x}$ and $F_{s,y}$ can be equal to zero. $F_{s,xy}$ becomes zero, when either $F_{s,x}$ or $F_{s,y}$ is equal to zero, and all un-zoomed node becomes points in a VZ layout, as shown in Figure A.1(b). This layout would confuse the user. Therefore, the second requirement, $F_{s,xy} > 0$, is required (as a demagnification factor, $F_{s,xy}$ is always less than 1).

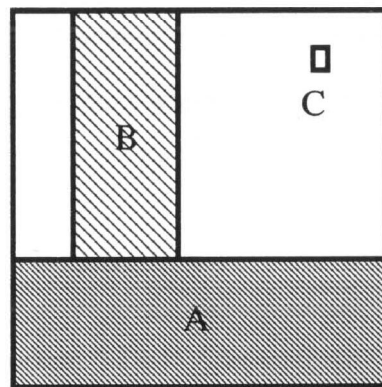


(a) The normal layout before a basic-operation.

(b) Zoomed node 'A' includes un-zoomed node 'B' and 'C' in the x direction, $F_{s,x} = 0$ and $F_{s,y} > 0$, $F_{s,xy} = \min(F_{s,x}, F_{s,y}) = 0$.



(c) Zoomed node 'A' includes un-zoomed node 'B' and 'C' in the x direction, $F_{s,x} = 0$ and $F_{s,y} > 0$, $F_{s,xy} = \max(F_{s,x}, F_{s,y}) = F_{s,y}$.



(d) Zoomed node 'A' and 'B' include un-zoomed 'C' in both x and y directions, $F_{s,x} = F_{s,y} = 0$; $F_{s,xy} = 0.2$.

Figure A.1: Calculating $F_{s,xy}$ when $F_{s,x} = 0$ and/or $F_{s,y} = 0$.
 ((a) is the layout of Current-nodes before a basic-operation, and (b) to (d) are the layouts after the basic-operations)

We observe that $F_s = 0$ is equivalent to saying that the zoomed nodes **include** all of the un-zoomed nodes¹. Consider zooming ‘A’ in Figure A.1(a); node ‘A’ includes all unzoomed nodes (nodes ‘B’ and ‘C’) in the x direction, and $F_{s,x} = 0$. If we want to zoom ‘A’ and ‘B’ together, they include all unzoomed nodes (node ‘C’) in both x and y directions, and $F_{s,x} = F_{s,y} = 0$. We also observe that an un-zoomed node, if scaled by a factor less than 1 (on an given axis) and included in the zoomed nodes (in this axis), will remain inside its Parent-node (in this axis) after a basic-operation, such as node ‘C’ in Figure A.1(b). In other words, in this case, any $F_{s,xy}$ less than 1 will make un-zoomed node ‘C’ meet the “within Parent-node” requirement on the x axis.

If $F_{s,x} = 0$ and $F_{s,y} > 0$, we can simply set $F_{s,xy} = F_{s,y}$ to satisfy both requirements of $F_{s,xy}$: “within Parent-node” and greater than zero. This is because (1) $F_{s,xy} > 0$ directly; (2) $F_{s,xy} = F_{s,y}$ guarantees all un-zoomed nodes are within the Parent-node in the y direction; and (3) in x direction, as described above, $F_{s,x} = 0$ indicates that all un-zoomed node must be included in the zoomed nodes, and such un-zoomed nodes satisfy the “within Parent-node” requirement when scaled by a factor less than 1 ($F_{s,y}$ is always less than 1). The result is shown in Figure A.1(c). When $F_{s,y} = 0$ and $F_{s,x} > 0$, the algorithm is identical.

¹The concept of inclusion is described in Subsection 3.3.1.1. The mathematical expression for zoomed nodes including all un-zoomed nodes is

$$r_2 = 1 \text{ or } S_z = S_a \quad (\text{A.2})$$

It is not difficult to derive Equation (A.2) from $F_s = 0$ using the equations described in Chapter 3, and *vice versa*.

When $F_{s,x} = F_{s,y} = 0$, all un-zoomed nodes are included in the zoomed nodes in both x and y directions. Therefore, $F_{s,xy}$ can be any value (k) between 0 and 1 to satisfy both requirements of $F_{s,xy}$. However, we further expect that, from Figure A.1(c) to (d), the un-zoomed node ‘C’ becomes smaller (at least, does not become much larger) when both node ‘A’ and ‘B’ are zoomed than only ‘A’ is zoomed. But, on the other hand, it does not seem worth developing a complex algorithm to adjust k with the environment and the user’s request to exactly meet this expectancy, because (a) the case $F_{s,x} = F_{s,y} = 0$ happens seldom in real applications, and (b) even if it happens, the sizes of un-zoomed nodes are not very important to the user, as long as they are small enough and meet both requirements of $F_{s,xy}$. As a compromising result, we set k to some small constant, such as 0.2, to make the algorithm simple and also meet above expectancy in a large degree. $F_{s,xy} = 0.2$ make node ‘C’ become smaller in Figure A.1(d) than (c) in most case; and even in the “bad” case, ‘C’ in Figure A.1(d) cannot be much larger than ‘C’ in (c).

Concluding the above discussion, we calculate $F_{s,xy}$, as follows:

```

IF       $minimum(F_{s,x}, F_{s,y}) > 0$ 
THEN    $F_{s,xy} = minimum(F_{s,x}, F_{s,y});$ 
IF       $(minimum(F_{s,x}, F_{s,y}) = 0) AND (maximum(F_{s,x}, F_{s,y}) > 0)$ 
THEN    $F_{s,xy} = maximum(F_{s,x}, F_{s,y});$ 
IF       $maximum(F_{s,x}, F_{s,y}) = 0$ 
THEN    $F_{s,xy} = 0.2;$ 

```

Appendix B

DERIVATION OF K_b FORMULA

This appendix derives Equation (3.18) used in the variable balance factor method, Section 3.5.

B.1 Definitions and Assumptions

Assume that the one-dimensional window size is L_w , and the detailed view is at the n^{th} level. We introduce following parameters for i^{th} level ($i = 1 \dots n-1$) :

- $K_{b,i}$ is the balance factor. As described in Subsection 3.3.5, under a given network configuration and user's requirement, $K_{b,i}$ determines the ratio of i^{th} level context with (all deeper level contexts plus detail).

- $R_{z,i}$ is the ratio between the sum of enlarging segments and the length of their immediate Parent-node after an basic-operation (refer to Equation (3.5)).
- $L_{e,i}$ is the sum of all enlarging segments after an basic-operation. For a single zoomed node:

$$L_{e,i} = L_{e,(i-1)} \cdot R_{z,i} \quad (\text{B.1})$$

and we define:

$$L_{e,0} \equiv L_w \quad (\text{B.2})$$

- $L_{s,i}$ is the sum of all shrinking segments after an basic-operation. From the one-dimensional point of view, $L_{s,i}$ is also the size of the context. The sum of all shrinking segments is equal to the difference of its immediate Parent-node and the sum of all enlarging segments.

$$L_{s,i} = L_{e,(i-1)} - L_{e,i} \quad (\text{B.3})$$

For a single zoomed node,

$$L_{s,i} = L_{e,(i-1)} - L_{e,i} = L_{e,(i-1)} \cdot (1 - R_{z,i}) \quad (\text{B.4})$$

In following derivations, we assume:

1. In each level, only one node is zoomed in.
2. At each level, r_4 is small enough to let (refer to Equation (3.13)):

$$R_{z,i} \approx K_{b,i} \quad (\text{B.5})$$

3. At the $(n - 1)^{th}$ level, the one-dimensional size of the context is equal to or smaller than that of the detail, i.e.

$$L_{s,(n-1)} \leq L_{e,(n-1)} \quad (\text{B.6})$$

4. The one-dimensional sizes of higher levels of context are less than or equal to that of lower levels, i.e.

$$L_{s,1} \leq L_{s,2} \leq \dots \leq L_{s,(n-1)} \quad (\text{B.7})$$

B.2 Derivation

Under above assumptions, Equation (B.1) becomes:

$$L_{e,i} \approx L_{e,(i-1)} \cdot K_{b,i} \approx L_w \cdot K_{b,1} \cdot K_{b,2} \cdot \dots \cdot K_{b,i}; \quad (\text{B.8})$$

and Equation (B.4) becomes

$$L_{s,i} \approx L_{e,(i-1)} \cdot (1 - K_{b,i}) \approx L_w \cdot K_{b,1} \cdot K_{b,2} \cdot \dots \cdot K_{b,(i-1)} \cdot (1 - K_{b,i}) \quad (\text{B.9})$$

Substitute Equation (B.8) and Equation (B.9) in Equation (B.6), to obtain, at the level $(n-1)$:

$$1 - K_{b,(n-1)} \leq K_{b,(n-1)} \quad (\text{B.10})$$

or

$$K_{b,(n-1)} \geq \frac{1}{2} \quad (\text{B.11})$$

Combining Equation (B.9) and Equation (B.7), we obtain:

$$1 - K_{b,1} \leq K_{b,1} \cdot (1 - K_{b,2}) \leq \dots \leq K_{b,1} \cdot K_{b,2} \cdot \dots \cdot K_{b,(n-2)} \cdot (1 - K_{b,(n-1)}) \quad (\text{B.12})$$

or

$$K_{b,i} \geq \frac{1}{2 - K_{b,(i+1)}} \quad (\text{B.13})$$

From Equation (B.11) and Equation (B.13), we can obtain the value of $K_{b,i}$ as¹:

¹We can prove Equation (B.17) by induction:

Assume $i = n - j$; ($j = 1, 2, \dots, n - 1$), Equation (B.13) becomes:

$$K_{b,(n-j)} \geq \frac{1}{2 - K_{b,(n-j+1)}} \quad (\text{B.14})$$

and Equation (B.17), which we want to prove, becomes:

$$K_{b,(n-j)} \geq \frac{j}{j+1} \quad (\text{B.15})$$

1. Equation (B.15) is true for $j = 1$, because Equation (B.11).
2. If Equation (B.15) is true for $j = m$, it will be true for $j = m+1$. Substituting Equation (B.15) in Equation (B.14), we have:

$$K_{b,(n-(m+1))} \geq \frac{1}{2 - K_{b,(n-m)}} \geq \frac{1}{2 - \frac{m}{m+1}} = \frac{(m+1)}{(m+1)+1} \quad (\text{B.16})$$

$$K_{b,i} \geq \frac{(n-i)}{(n-i)+1} \quad (i = 1 \dots n-1) \quad (\text{B.17})$$

Here, $(n-i)$ is the distance from the i^{th} level of context to the n^{th} level of detail. $K_{b,i}$ increases with the distance; in other words, the size of a level of context decreases with its distance from the detail. Equation (B.17) also keeps $K_{b,i}$ greater than or equal to 0.5, and less than 1.

B.3 Result

K_b is a user-controllable parameter. The value of K_b at each level essentially depends on user requirements and network configuration. It is difficult to determine an optimal value of K_b for a given application in advance. Therefore, after the derivation, we could ignore all the assumptions mentioned before and use the following simple equation as suggested values to assist the user in specifying $K_{b,i}$.

$$\mathbf{K}_{b,i} = \frac{(\mathbf{n} - \mathbf{i})}{(\mathbf{n} - \mathbf{i}) + 1} \quad (\mathbf{i} = 1 \dots \mathbf{n} - 1) \quad (\text{B.18})$$

As defined before, $K_{b,i}$ is the balance factor at the i^{th} level, when the detail view is at the n^{th} level.

REFERENCES

- [Beard 90] Beard D.V., and Walker, J.Q., Navigational Technique to Improve the Display of Large Two-dimensional Space, *Behavior and Information Technology*, Vol. 9, No 6, pp.451-466, 1990.
- [Becher 91] Becher, R.A., Eich, S.G., Wilks, A.R., Basics of Network Visualization, *IEEE Computer Graphics and Application*, pp.12-14, May 1991.
- [Brooks 86] Brooks F.P., Walkthrough - a Dynamic Graphics System for Simulation Virtual Buildings, *Proceeding of the 1986 Workshop on Interactive 3D Graphics*, pp.23-24, Chapel Hill NC 9-21, October 1986.
- [Card 91] Card, S.K., Robertson, G.G., Mackinlay, J.D., The Information Visualizer, an Information Workspace, *CHI'91*, pp.181-188, 1991.
- [Crawford 89] Crawford, J.L., Graphics for Network Management: an Interactive Approach, *Integrate Network Mangement*, I. B. Meandzija and J. Westcott (editors) Elsevier Science Publishers B.V., pp.197-208, 1989.
- [Delisle 86] Delisle, N. and Schwartz, M., Neptune: a Hypertext System for CAD Applications, *Proceedings of ACM SIGMOD International Conference on Management of Data*, 28-30 May, Washington, DC, pp.132-143, 1986.
- [Fairchild 88] Fairchild, K.M., Poltrock, S.E., Furnas, G., W. SemNet: Three-dimensional Graphic Representation of Large Knowledge Bases, *Cognitive Science and its Application for Human-Computer Interface*, R. Guindon (Editor). Lawrence Erlbaum, New Jersey, pp.201-233, 1988.
- [Foley 90] Foley, J.D., van Dam, A., Feiner, S.K. and Hughes, J.F., (second edition) *Computer Graphics Principles and Practice*, Addison-Wesley Publishing Company, 1990.

- [Furnas 86] Furnas, G.W., Generalized Fisheye Views, *Proc. Human Factors in Computer System, CHI'86, ACM*, pp.16-23, Boston, MA, April, 1986.
- [Gentner 83] Gentner, D., and Stevens, A.L., editors, *Mental Models*, Erlbaum, Hillsdale, N.J., 1983.
- [Halasz 87] Halasz, F.G., Moran, T.P. and Trigg, R.H., NoteCards in a Nutshell, *CHI'87 and GI Proceedings*, April, pp.45-52, 1987.
- [Havens 91] Havens, W., Dill, J., Dickinson, J., Crawford, J., Begg, I., Kheraj, R., Moore, W., Intelligent Graphics Interfaces for Supervisory Control and other Real-Time Database Applications, *Proc. 7th Int. conf. on Data Eng.*, pp.8 -12, April 1991.
- [Henry 91] Henry, T.R., Hudson, S.E., Interactive Graph Layout, *UIST'91 Nov.*, p55, 1991.
- [Hollands 89] Hollands, J.G. Carey, T.T., Matthews, M.L., McCann, C.A, Presenting a Graphical Network: A Comparison of Performance Using Fisheye and Scrolling Views, in *Designing and Using Human-Computer Interfaces and Knowledge Based Systems*, edited by G. Salvendy, 1989.
- [Hollnagel 83] Hollnagel, E., Woods, D., Cognitive systems engineering: New Wine in Old Bottles, *International Journal of Man-Machine Studies*, 18, pp.583-600, 1983.
- [IGI 92-0087] Begg, I., IGI Proof-of-Concept Prototype 1 System Requirements Specification - Issue 1, *IGI Technical Report 92-0087*, September 1992.
- [IGI 92-0071] Zuo, Z.P., and Dill, J., Variable Zoom: A New Display Technique for Large Hierarchical Network, *IGI Technical Report 92-0071*, July, 1992.
- [IGI 92-0061] Dubs, S., Roseman, M., Schaffer, D., Comparing Fisheye and Full-Zoom Techniques for Navigation of Hierarchical Graphs, *IGI Technical Report 92-0061*, July 1992.
- [IGI 92-0046] Bartram, L., J. Dill., G/UI Candidate Techniques, *IGI Technical Report 92-0046*, May 1992.
- [IGI 90] *Intelligent Graphic Interface Project Feasibility Study Report*, Microtel Pasific Research Limited, Simon Fraser University, SF Univentures, and B.C. Advanced Systems Institute, March, 1990.

- [IGI 89] *Intelligent Graphic Interface Project Feasibility Study Interim Technique Report*, Microtel Pasific Research Limited, Simon Fraser University, SF Univentures, and B.C. Advanced Systems Institute, 1989.
- [Jacks 68] Jacks, E.L., A Laboratory for the Study of Graphical Man-machine Communications, *Joint Computer Conf., AFIPS Conf. Proc.*, vol. 26, Montrale, N.J., AFIPS Press. pp.363-386, 1968.
- [Kantowitz 83] Kantowitz, B.H., Sorkin, R.D., *Human Information Processing*, Wiley, New York, 1983.
- [Mackinlay 91] Mackinlay, J.D., Robertson, G.G., and Cart, S.K., The Perspective Wall: Detail and Context Smoothly Integrated, *CHI'91 Proceedings*, pp.173 - 179, April 1991.
- [Marcus 91] Marcus, A., and van Dam, A., User-Interface Developments for the Nineties, *IEEE Computer*, pp.49 - 57, September 1991.
- [Meunier 88] Meunier, J.M., An Interactive Network Display System for Network Management, *Proc. of IEEE 1988, Network Operation and Management Symposium*, February 1988.
- [Miller 56] Miller, G.A., The Magic Number Seven, Plus or Minus Two, *Psychological Review*, 63, pp.81-97, 1956.
- [Mitchell 87] Mitchell, C.M., and Saisi, D.L., Use of Model-based Qualitative Icons and Adaptive Windows in Workstations for Supervisory Control Systems, *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17(4), pp.573-593, July/August, 1987.
- [Mitta 89] Mitta, D.A., Fisheye Representation of Information: IMIS User Interface, (Technical Report Air Force Office of Scientific Research/AFSC, 1989 USAF-UES Summer Faculty Research Program, Contract F49620-88-C-0053), Wright-Patterson Air Force Base, OH: Human Resources Laboratory, September 1989.
- [Mor 81] Moray, N., The Role of Attention in the Detection of Errors and the Diagnosis of Errors in Man-machine System, In Rasmussen, J and Rouse, W.B., editors, *Human Detection and Diagnosis of System Failures*, pp.185-198, Plenum, New York, 1981.
- [Newman 68] Newman, W. M., *A System for Interactive Graphical Programming*, Thompson Books, Washington, D.C., pp.47-54, SJCC 1968.

- [Newell 90] Newell, A., *Unified Theories of Cognition*, Harvard University Press, 1990.
- [Ramanathan 89] Ramanathan, J., Hartung, R.L., A Generic Iconic Tool for Viewing Databases, *IEEE Software*, pp.50-57, September 1989.
- [Resnikoff 89] Resnikoff, H.L., *The Illusion of Reality*, Spring -Verlag, New York, 1989.
- [Robertson 91a] Robertson, G.G., Mackinlay, J.D, and Card, S.K., Information Visualization Using 3D Interactive Animation, *CHI'91 Proceedings*, pp.461 - 462, 1991.
- [Robertson 91b] Robertson, G.G., Mackinlay, J.D, and Card, S.K., Cone Trees: Animated 3D Visualizations of Hierarchical Information, *CHI'91 Proceedings*, pp.189-194, 1991.
- [Sarkar 92] Sarkar, M., Brown, M.H., Graphical Fisheye Views of Graphs, *CHI'92 Proceedings*, pp.83-91, May, 1992.
- [Schaffer 92] Schaffer, D., Zuo, Z., Bartram, L., Dill, J., Dubs, S., Greenberg, S., and Roseman, M., Comparing Fisheye and Full-zoom Techniques for Navigation of Hierarchically Clustered Networks, *Research Report No. 92/491/29, Dept. of Computer Science of the University of Calgary*, (has been submitted to GI 93), November, 1992.
- [Spence 82] Spence, R., and Apperley, M., Data base Navigation: An Office Environment for the Professional, *Behavior and Information Technology* 1 (1), pp.43-54, 1982.
- [Wickens 84] Wickens, C.D., *Engineering Psychology and Human Performance*, Merrill Publishing, Columbus, OH, 1984.