An Evaluation of Content Browsing Techniques for Hierarchical Space-Filling Visualizations

Kang Shi*, Pourang Irani†, Ben Li‡
Department of Computer Science
University of Manitoba, Canada

ABSTRACT

Space-filling visualizations, such as the TreeMap, are well-suited for displaying the properties of nodes in hierarchies. To browse the contents of the hierarchy, the primary mode of interaction is by drilling-down through many successive layers. In this paper we introduce a distortion algorithm based on fisheye and continuous zooming techniques for browsing data in the TreeMap representation. The motivation behind the distortion approach is for assisting users to rapidly browse information displayed in the TreeMap without opening successive layers of the hierarchy. Two experiments were conducted to evaluate the new approach. In the first experiment (N=20) the distortion approach is compared to the drill-down method. Results show that subjects are quicker and more accurate in locating targets of interest using the distortion method. The second experiment (N=12) evaluates the effectiveness of the two approaches in a task requiring context, we define as the context browsing task. The results show that subjects are quicker and more accurate in locating targets with the distortion technique in the context browsing task.

Keywords: browsing, distortion, hierarchy navigation, focus+context, drill-down, space-filling visualization, TreeMap, semantic zooming.

1. INTRODUCTION

Hierarchical data structures are regularly used interactively. They describe the relationships among entities in organizations, in file systems, and in family genealogies. The most common form of hierarchical representation is a node-link tree. However, trees are difficult to browse and are not space efficient. A significant amount of space remains unused in the background as a result of creating an adequate layout for the nodes.

Space-filling visualizations have been developed to make more efficient use of the display area. These systems are characterized by their compactness and effectiveness at showing the properties of nodes in a hierarchy through size or color. An example of a space-filling technique is the TreeMap [14]. In the TreeMap, the display is divided into rectangular regions to map an entire hierarchy of nodes and their children (Figure 1). Each node uses an amount of space relative to the weight of the item being represented in the hierarchy (such as the relative size of file in directories or the volume of shares sold on the stock market).

TreeMaps are well-suited for revealing global patterns in the data such as large pockets of empty space on a disk drive. However, the standard browsing mechanisms provided for inspecting the data can be complex, in particular as the size of the hierarchy grows larger. The method utilized by the TreeMap for browsing data is through *drilling down* into (moving down) the hierarchy or *rolling up* (moving up) to find nodes of interest. This interaction approach is very common and has been widely established by file and directory explorers provided in most current operating systems. The typical user interaction for locating a node consists of clicking the parent directory (or subtree) in which might reside a node of interest. The subtree fills then entire space and the user can recursively select subtrees until reaching their final location or node (Figure 1). This form of interaction is analogous to zooming into a region of interest with each step of the zoom operation being a subtree in the hierarchy. In general, and particularly in the context of this paper, content browsing refers to the task of locating specific content, such as a photograph or document. This is analogous to flipping through pages of a catalog of products or a phone directory.

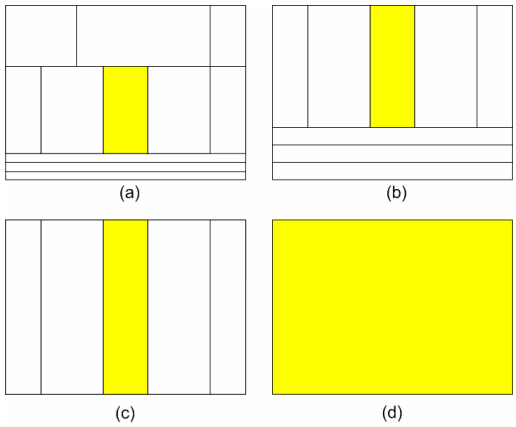

Figure 1: TreeMap uses a drill-down approach to open successive layers of a hierarchy. Selecting a parent node opens it and presents the subtree in the entire view. A total number of four drill-down operations are required to inspect the node highlighted above.

Several problems arise with the standard navigation scheme of drilling down and rolling up. Users can spend a significant amount of time browsing for specific items in hierarchies using such an interaction. In space-filling visualizations the only visual cue available to the user for locating a node are its visual attributes, such as color and size of node. This reduces the user's ability to quickly find elements unless they can adequately match the object sought after to its visual mappings. Another drawback with the drill-down approach is the number of unnecessary "trips"

*kangshi@cs.umanitoba.ca
†irani@cs.umanitoba.ca
‡lipak@cs.umanitoba.ca

IEEE Symposium on Information Visualization 2005
October 23-25, Minneapolis, MN, USA
0-7803-9464-X/05/$20.00 ©2005 IEEE.

a user may take to reach the file adequately. In the drill-down approach, traversing each successive layer requires abandoning the previous view. This can typically lead to disorientation during navigation and reduce the amount of context available for the task. The lack of context in browsing can negatively impact performance as the user has to internally reorient and reestablish relations between views to determine the group or cluster to which an element belongs to.

In this paper we introduce a browsing technique as an alternative to the drill-down approach used in TreeMaps by using fisheye and semantic zooming techniques. With this approach, a node of interest may be inspected from the highest level of the hierarchy. Given that in space-filling representations, all nodes are visible (or at least as much as possible) from the highest level view, the new technique seems very conducive for browsing items. There are also potential problems with the distortion technique over the drill-down approach. In the drill-down approach the visibility of nodes increases when a directory is opened or zoomed into. However, using the distortion technique, nodes that are not clearly visible on the surface of the hierarchy could be difficult to expand. Additionally, we are interested in determining whether the focus+context technique enhances performance in tasks requiring visual comprehension of the context. In light of these questions, we conducted two experiments to evaluate the performance of the distortion technique under controlled conditions.

2. RELATED WORK

Three general categories of work are directly related to the work we present in this paper. The first set of research consists of applications of focus+context techniques for browsing hierarchy structures presented using space filling visualizations. The second category of work relates to techniques that have been developed specifically for browsing data in the TreeMap. The third category of work consists of results under the collection of continuous semantic zooming techqniues, which has inspired this work.

2.1 Space-Filling Focus+Context Representations

Focus+Context techniques assist users in viewing peripheral information while maintaining their focus on the elements of interest. They have been applied to the display of graphs [7], trees [12], and tabular data [11]. To the best of our knowledge, such techniques have not been applied to the TreeMap. However, several space-filling visualizations have taken advantage of these. Stasko and Zhang [15] designed a radial space-filling technique, called Sunburst. In the Sunburst technique the root of the hierarchy is placed in the center of the visual space and files and directories are laid out radially in wedges extending from the center. Each level of the hierarchy is a concentric circle and the deepest level is furthest away from the center. The size of the file or directory is represented by the angular sweep of each wedge.

Three focus+context displays were designed for the Sunburst: angular detail, detail outside and detail inside. These techniques allow users to interact with subtrees in the hierarchy by facilitating the view of small items in detail while providing context of the entire hierarchical structure. Each technique takes advantage of smooth animated transitions between the views to help users maintain their orientation during navigation tasks. While each of these techniques is well suited for showing detail and overview of the hierarchy structure, they are not created with the intention of viewing the content of nodes within hierarchies.

Another example of focus+context for space-filling visualizations is the Information-Slices technique for displaying the details of substructures within a hierarchy [1]. As the user clicks on a node, its representative sub-structure is opened using a semi-circle presentation and linked by an arc. As in the focus+context techniques for the Sunburst, the semi-circle is designed primarily to facilitate detail viewing of substructure within a hierarchy.

In both the visualizations described above, focus is created by "fanning-out" and enlarging the subtree of interest. Context is provided by presenting the original hierarchy in the same view and by visually depicting the relation between the subtree and its location in the original hierarchy. Radial space-filling visualizations have the added advantage of being capable of showing the hierarchical structure more explicitly than other space-filling representations such as the TreeMap. Therefore replicating the same type of focus+context techniques onto the TreeMap may not lead to a visually comprehensible presentation.

2.2 Browsing Techniques Applied to the TreeMap

In addition to focus+context methods, several other interaction techniques assist in browsing information structures. Brushing for example, is an effective technique for browsing information and performing exploratory data analysis. In particular, brushing assists the user in selectively isolating subsets of data for exploration and inspection. Fua et al. [6] designed an interactive structure-based brushing technique which can be used to perform selection in hierarchical datasets. Structure-based brushing was applied to TreeMaps to assist users in selecting clusters of interest using a structure-based coloring [6]. Using this tool, users can specify regions of interest based on the location or depth of the clusters of interest in the hierarchy. By dynamically selecting a bounding region in the structure-based brush, corresponding elements in the TreeMap get highlighted. Using such a technique clusters of interest (based on the variable being mapped in the display) can be selectively inspected. This technique however, does not display the content of nodes in the TreeMap.

To the best of our knowledge, the primary variation of the TreeMap that has facilitated direct browsing of hierarchical content is the Quantum TreeMap[5]. Quantum TreeMaps were designed to facilitate browsing of entities in a hierarchy that consist of 'quantum' or indivisible size, such as images. An integral component of the quantum TreeMap is an ordered layout algorithm which maximizes the amount of space available for the nodes in the hierarchy by rearranging the display based on the size of the elements.

Photomesa [2], an application based on the Quantum TreeMap, displays a thumbnail of all images in a directory. The basic mechanism for browsing images or other content is achieved by hovering or zooming into thumbnails of interest. Smooth animation between different endpoints in the zooming operation facilitates context viewing. However, the objects zoomed into overlap the Quantum TreeMap and occlude parts of the display region. Furthermore, to navigate or browse from one region to another the user needs to roll out and drill back into the area of interest. Another characteristic of the Quantum TreeMap is that the underlying hierarchical structure of the information is collapsed onto a flattened view to facilitate the task of browsing. The motivation to flatten the hierarchical structure is based on the assumption that users are typically interested in groups of items and not the inter-structural relationships between these.

In both the techniques described above, drilling down into levels of the hierarchy is the prevalent form of navigation. Brushing provides a mechanism for selecting and filtering items of interest based on its visual property (such as color mapping) or depth in hierarchy. In the case of the Quantum TreeMaps, hierarchical structures with elements of heterogeneous sizes and content may not be easily browse-able. An implicit requirement for the technique we introduce is to allow users to investigate node content in the tree without any transformation on the underlying structure.

2.3 Continuous Semantic Zooming

Our work is primarily inspired by the concept of continuous semantic zooming (CSZ) developed by Schaffer et al [13]. This technique is characterized by two distinct but interrelated components: continuous zooming [4] and presentations of semantic content at various stages of the zoom operation. CSZ manages a 2D display by recursively breaking it up into smaller areas. A region of interest becomes the focus and as the continuous zoom is applied, successive layers of a display "open up" (Figure 2). At each level of the operation the technique enhances continuity through smooth transitions between views and maintains location constraints to reduce the user's sense of spatial disorientation. The amount of detail shown in parts of the display is controlled by pruning the display and presenting items of non interest in summary form. A study comparing continuous semantic zooming to drill-down (full zoom), on a network of hierarchical clusters, shows that users can navigate and perform tasks related to node-link diagrams more efficiently with CSZ than with the traditional approach.

Continuous semantic zooming has been applied to information structures other than topological graphs. Datelens [3] employs continuous semantic zooming to reveal varying degrees of content in tabular structures in a smooth and continuous manner. The distortion in Datelens is linear and is applied to the cells of interest in a grid. As the level of distortion increases semantic information is revealed based on the size of the region available for the display. An evaluation comparing Datelens to common calendar based interactions reveals that continuous semantic zooming enhances content browsing in tabular structures [3]. The distortion algorithm used in Datelens cannot be directly applied to the TreeMap as the alignment of cells in the TreeMap is not symmetrical. Furthermore, TreeMap uses a hierarchical and not a tabular structure.

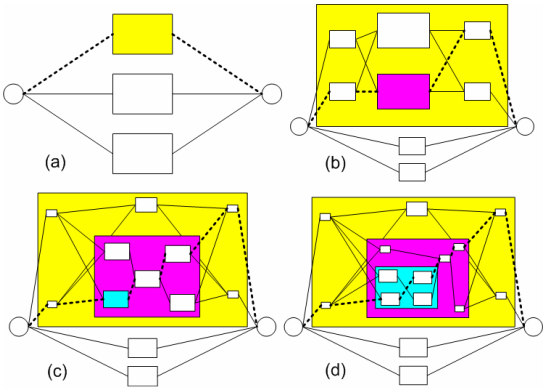

Figure 2: Continuous Semantic Zooming applied to a clustered network. Details of each cluster are visible based on the level of zooming employed. Progressing from (a) to (d) reveals more details as smooth transitions are created between views [13].

The approach we adopt is similar in concept to the continuous semantic zoom: smooth transition between zoom levels is applied and content visibility is increased as the nodes enlarge.

3. ALGORITHM

The distortion algorithm increases the size of a node of interest while shrinking its neighbors. Figure 3 depicts the behavior of the distortion algorithm as the user clicks on the node of interest. The node grows as the mouse selection is maintained and returns to its original size upon release of the mouse selection. The contents are revealed gradually as the node grows in size. Instead of presenting only a subset of the tree during the exploration operation (as is the case with the drill-down), in this approach the user can continuously select items from any location in the hierarchy and inspect their contents. Traversing layers of the hierarchy is thereby removed. However, distortion technique does not work for small or invisible nodes which cannot be located by the user. In this section we describe the data structure and algorithms for the distortion technique.

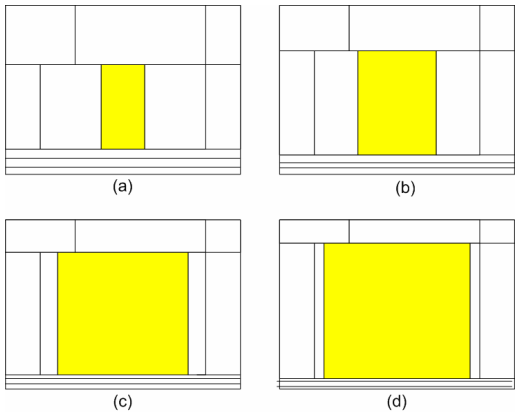

Figure 3: Distortion can be applied to nodes of interest. As the node expands, content data is revealed. From rest (a) to full expansion of a selected node (d).

3.1 Data Structures

Each displayable item in the TreeMap is represented as a node in the tree. The algorithm for displaying a TreeMap is well-known and described in [14]. Each node must contain enough information so that it can be redrawn as needed. In addition, each node must contain additional information, which will enable detection of neighbors and proper distortion of nodes.

We define a node to contain the following information:

- *size (weight):* indicates the size of a node. For a data file, this may be the size of a file. For a directory, this may be the cumulative size of all the files contained within it.
- *width, height:* the width and height of the node when it is drawn on the screen. These values are computed based on its size and the size of its parent node.
- *orientation:* this field determines whether the node is displayed horizontally or vertically with respect to its siblings in the TreeMap. Nodes at the same level have the same orientation, and the orientation alternates between levels of the TreeMap.
- *parent:* this field is a pointer to the parent node.
- *prevSibling, nextSibling:* these are pointers to the node's previous and next siblings. A node's previous sibling is the

sibling that is drawn to the left of it, if the orientation is horizontal and is the sibling that is drawn above it, if the orientation is vertical. A node's next sibling is the sibling that is drawn to the right of it, if the orientation is horizontal and is the sibling that is drawn below it, if the orientation is vertical.

- *children:* this is an array of pointers to each of its children.
- *leftNeighbor:* this pointer is set to its left sibling (in the TreeMap) which contains the node of interest, if this is the "right-neighbor" of the node of interest. Otherwise it is set to NULL. The *rightNeighbor, topNeighbor, bottomNeighbor* member variables are defined in a similar manner. This variable is described in more detail in the next section.
- *amount:* the increment of distortion for each step.

In addition to the data structure for a node, there are several other important data items in the distortion algorithm. The root of the tree is denoted by *ROOT*. The constant *MIN_SIZE* will denote the smallest width or height that a node can shrink to. The constant *MAX_SIZE* denotes the largest width or height that a node can grow to. Finally, we have four global variables *gLeftNeighbor, gRightNeighbor, gTopNeighbor, gBottomNeighbor* which denote the left, right, top and bottom neighbors of the node of interest. These four variables represent the nodes that will shrink to make room for the node of interest.

3.2 Computing Neighbors

When a node A has been selected as the "node of interest", that is, the node that is to be distorted, the first step of the distortion process is to compute the neighbors of A. A *neighbor* of a node A is any other node B such that:

1. A and B have overlapping borders in the TreeMap,
2. B is not an ancestor of A, and
3. B is as near the root node as possible.

Suppose A has a *left-neighbor* B; that is, B is a neighbor of A where B's right border and A's left border intersect. Furthermore, suppose C is some other node where C's right border and A's left border intersect and C is not an ancestor of A. Then, by the above definition, it can easily be seen C must be an ancestor of B. This is similarly true for the *right-neighbor*, *top-neighbor* and *bottom-neighbor* of A, which are defined similarly as *left-neighbor*. Thus, using this definition, any node A has at most one *left-neighbor*, *right-neighbor*, *top-neighbor*, and *bottom-neighbor*.

Suppose that B is the left-neighbor of A, it is useful to keep track of the ancestor C of A such that B and C are at the same level in the tree, which is stored in *B.rightNeighbor*. Clearly, B and C are siblings. This information is useful when applying the distortion algorithm, which is explained below. It is easy to see that C must be *B.nextSibling*. Note that it is possible for A and B to be at the same level, in which case C is A, as *B.nextSibling* is A. Similarly, we can define the right-neighbor, top-neighbor and bottom-neighbor. It should be noted that the four neighbors of a node A must be distinct and it is possible for node A to not have a neighbor.

The following algorithm computes the left-neighbor of a node A. Since the left-neighbor of A is to the left of A, we check to see if *A.PrevSibling* is *NULL* only when the orientation is horizontal. The algorithm for computing the right, top and bottom neighbors may be specified in a similar manner.

Algorithm 3.2.1: COMPUTELEFTNEIGHBOR(*A*)

```
local LeftNeighbor
if A ≠ ROOT
then { if A.orientation = HORIZ and A.PrevSibling ≠ NULL
       then { LeftNeighbor ← A.PrevSibling
              LeftNeighbor.RightNeighbor ← A
              return (LeftNeighbor)
       else return (COMPUTELEFTNEIGHBOR(A.parent))
else { LeftNeighbor ← NULL
       return (LeftNeighbor)
```

When *ComputeLeftNeighbor* is called with the "node of interest", say A, the algorithm will determine its left-neighbor B such that A's left border intersects B's right border. The routine will return B, and *B.RightNeighbor* is set to *B.nextSibling*, which is A, if A and B are at the same level, or else an ancestor of A.

3.3 Changing Node Size

When the size of a node is modified, this needs to be propagated to all its children. The propagation algorithm is given below.

Algorithm 3.3.1: CHANGENODESIZE(A,δ)

```
n ← number of children of A
oldsize ← A.size
A.size ← A.size + δ
for i = 0 to n − 1
  do CHANGENODESIZE(A.children[i], (oldsize + δ)
                    * A.children[i].size / oldsize)
```

3.4 Distortion Algorithm

The following algorithm decreases the size of the left-neighbor of the node of interest. This left-neighbor is given by *gLeftNeighbor* and initially computed by *ComputeLeftNeighbor*. The algorithm decreases the size of the left-neighbor by *amount* and increases *gLeftNeighbor.rightNeighbor* by the same *amount*. This ensures that the overall size of the TreeMap is not changed, and the node of interest's size, which is a descendent of *gLeftNeighbor.rightNeighbor*, is increased.

Algorithm 3.4.1: DISTORTLEFT (*gLeftNeighbor, amount*)

```
global gLeftNeighbor
comment: Executes on iteration of the distortion algorithm
if gLeftNeighbor ≠ NULL
then { if gLeftNeighbor.width > MIN_WIDTH
       then { CHANGENODESIZE(gLeftNeighbor, −amount)
              CHANGENODESIZE(gLeftNeighbor.RightNeighbor,
                             amount)
       else { if gLeftNeighbor.prevSibling ≠ NULL
              then { temp ← gLeftNeighbor.RightNeighbor
                     gLeftNeighbor ← gLeftNeighbor.prevSibling
                     gLeftNeighbor.RightNeighbor ← temp
              else gLeftNeighbor ← COMPUTELEFTNEIGHBOR(
                                   gLeftNeighbor)
```

If node A is influenced by algorithm 3.4.1 such that its width is less than *MIN_WIDTH*, then propagation is applied. This involves computing the left-neighbor of node A. There are two possible scenarios. If A has a sibling to the left of it, then this sibling is now the left neighbor of the node of interest. Otherwise, we need to compute the left neighbor of this node A using the *ComputeLeftNeighbor* method. DistortRight, DistortTop and DistortBottom can be similarly defined.

The following algorithm describes the distortion technique. It begins by determining the neighbors of the node of interest, A. Then it decreases each of the sizes of these neighbors of A, while increasing the size of A which provides the distortion effect. Then the entire TreeMap is redrawn. This process is repeated until an external event stops the distortion process, or the node of interest being increased has reached a fixed maximum width or height.

Algorithm 3.4.2: DISTORTALGORITHM (*A*)

```
global gLeftNeighbor, gRightNeighbor
global gTopNeighbor, gBottomNeighbor
gLeftNeighbor ← COMPUTELEFTNEIGHBOR(A)
gRightNeighbor ← COMPUTE RIGHT NEIGHBOR(A)
gTopNeighbor ← COMPUTE TOP NEIGHBOR(A)
gBottomNeighbor ← COMPUTE BOTTOM NEIGHBOR(A)
while DISTORTING = true and A.width < MAX_SIZE
and A.height < MAX_SIZE
do { DISTORTLEFT (gLeftNeighbor, amount)
     DISTORTRIGHT (gRightNeighbor, amount)
     DISTORTTOP (gTopNeighbor, amount)
     DISTORTBOTTOM (gBottomNeighbor, amount)
     REDRAWTREEMAP (ROOT)
     SLEEP (sleep_interval)
```

Figure 4 shows the effect of clicking and releasing a node. The content appears as the node is being opened and the distortion appears gradually. The remainder of this paper describes the results of our evaluation.

4. EXPERIMENT 1 – BROWSING

Experiment 1 was designed to compare the Drill-Down method to the Distortion method for locating specific content (pictures) in the TreeMap. While our task is designed for inspecting images, the technique could be generalized to view other types of files, such as text files, movie files, etc. For this reason we have used the original TreeMap implementation instead of its alternatives such as the Quantum TreeMap or squarified TreeMap. We anticipated the following effects from our experimental data:

Hypothesis 1: Overall, participants will locate the target faster via the Distortion method versus the Drill-Down method.

Hypothesis 2: In deep hierarchical structures, participants will locate the target faster via the Distortion method versus the Drill-Down method.

Hypothesis 3: In the Distortion method performance will not differ between deep and wide hierarchies.

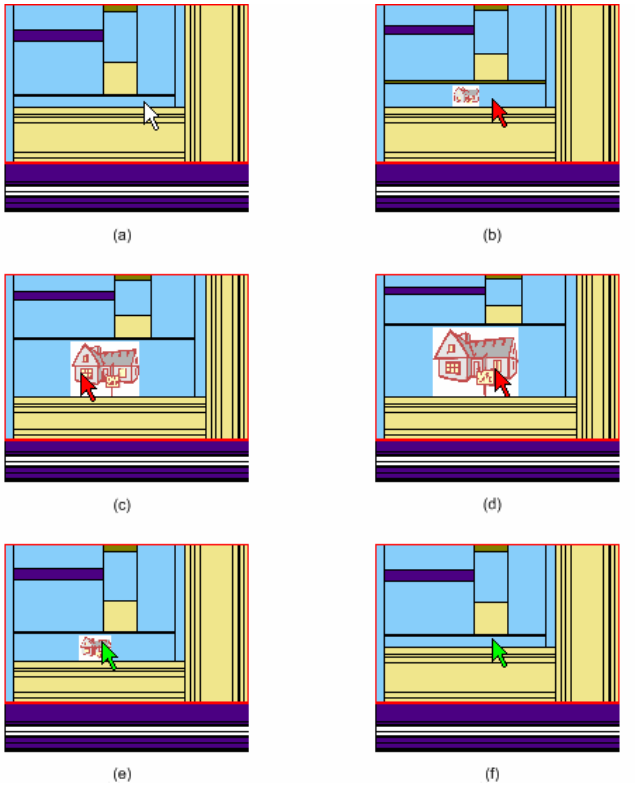

Figure 4: Implementation results of the algorithm described above. The red arrow indicates selection (b)-(d) and the green arrow indicates release of the mouse button (e) and (f).

4.1 Method

4.1.1 Subjects

Twenty undergraduate students participated in the experiment and were assigned to one of the two conditions: Distortion first or Drill-Down first. Subjects were volunteers from a computer science course in HCI. All were familiar with the concept of file and directory structures and had reasonable experience performing standard file management routines. None had any previous experience with the TreeMap although they were familiar with space-filling concepts and the TreeMap representation.

4.1.2 Materials

Two different types of hierarchy were used for the experiment: deep and wide. To maintain simplicity in the experimental procedure and to avoid invisible small nodes, we did not use very large data sets. The deep hierarchy was constructed using six levels, with a maximum of three sub-directories per node. The wide hierarchy was created with a depth of three levels, and each node contained a maximum of six sub-directories. Both types of hierarchies, deep and wide, contained thirty different pictures each and more than three hundred files of various other types. To reduce learning effects, we used two sets of hierarchies (Set A and Set B) which were created with the same hierarchical structures but entirely different images and files. Half the participants started the experiment with the Drill-Down method and the other half started using the Distortion method. After completing the tasks in one set of hierarchies with one method, the participants switched onto the other set of hierarchies with the

other method. All tasks in the experiment required that subjects locate a specific picture in the TreeMap.

Participants performed the experiment on a 17" monitor with a 1024x768 resolution. The prototype ran over Windows XP. The task was described to them before they began running the trials.

4.1.3 Procedure

Before starting the experiment, each subject got familiarized with both browsing techniques. The experiment started when the participant indicated that he or she was comfortable using the tool and its interface.

In each task, we randomly chose one picture as a target picture from all thirty pictures in a hierarchy, and displayed the target picture to the participant in a window outside of the TreeMap. Half of the image files used as targets were small and occupied only a small fraction of space on the display. The subject was required to browse the hierarchy until he/she located the target image in the TreeMap. The target image was available throughout the task.

Each participant performed 3 trials with wide hierarchies and 3 trials with deep hierarchies in the following sequence W1, D1, W2, D2, W3, and D3, where W represents the wide hierarchies and D represents the deep hierarchies. The participant was free to end the trial if they could not locate the specified picture. A time limit was not imposed for this task. We recorded whether the participant located the correct target, whether the participant withdrew from the task, and the time to execute the task in all conditions. In summary, the whole experiment involved: 20 participants × 2 main conditions × 2 types of hierarchies × 3 trials = 240 trials in total.

4.2 Results and Discussion

To test the three hypotheses stated in the beginning of this section, we measured subjects' performance on the given task with respect to time until completion. We recorded the average response for locating the target. Of 240 trials to locate the target, 13 attempts were incomplete. Of these 13 attempts to locate the target, 9 attempts resulted in giving up on the task. All 13 incomplete/give-up results were excluded from the data analysis.

Table 1: Average completion times for Wide and Deep hierarchies with both Methods (standard deviations are in parentheses).

	Wide	Deep
Distortion	31.6 (20.8) sec	26.64 (22.5) sec
Drill-Down	52 (33.8) sec	68.2 (47.1) sec

The results are summarized in Table 1. Average completion times were not consistent with the normality assumptions in both datasets (distortion or drill-down). The analysis was therefore performed on the log transform of the recorded performance times. The time to locate target data were analyzed by means of a 2×2 (Type of Method × Hierarchical Structure) one-way analysis of variance (ANOVA), with both Type of Method (Drill-Down vs. Distortion) and Hierarchical Structure (Deep vs. Wide) serving as repeated measures. An alpha level of .05 was used for all statistical tests. Type of Method was found to be significant ($F(1, 223) = 46.68$, $p < .001$) with the Distortion method group's mean task time (27.53 sec) being faster than the Drill-Down method group's (50.01 sec). The main effect for Hierarchical Structure was not statistically significant ($F(1, 223)=2.18$, $p = .14$). However, a significant interaction effect was found between Type of Method and Hierarchical Structure, $F(1, 223) = 9.06$, $p < .01$.

In conjunction with the means, it is clear that participants completed the task faster overall with the Distortion method vs. the Drill-Down method, regardless of conditions of Hierarchical Structure; this supports the first hypothesis. The significant interaction tells us that the effect of Type of Method depends on the level of Hierarchical Structure. The simple effect of Method for the wide hierarchical structure clearly indicates that a faster mean task time is achieved via the Distortion method than the Drill-Down method (31.6 seconds vs. 52 seconds). The simple effect of Method for the deep hierarchy is more pronounced (26.64 seconds vs. 68.2 seconds). The 95% confidence intervals for the simple effects of Method at both levels of Hierarchical Structure are significantly different. Taken together with the significant main effect found for Type of Method, there is strong support for the second hypothesis, i.e. the Distortion Method will be faster than the Drill-Down Method, especially in deep trees.

A close observation of mean completion times for the distortion technique reveals that on average subjects are faster with deep hierarchies than with wide hierarchies. A paired sample T-Test shows this difference in means to be non-significant ($T(1,59)=1.264$, $p=0.221$), supporting hypothesis 3, i.e. subjects will perform equally well on deep and wide hierarchies with the distortion technique.

We did not observe any differences in mean completion times for targets occupying a fraction of the display space. However, in certain cases we observed that participants ignore potential targets if these occupied a small amount of space. Less than 3.75% of the total number of trials consisted of participants missing small targets. In hierarchies with over thousand nodes this could potentially affect the performance of the distortion technique. A possible solution to alleviate this problem would consist of combining the drill-down to open a node (which increases the reserved space) followed by distortions.

5. Experiment 2 – Browsing with Context

Focus+context techniques are designed with the aim to facilitate navigation or browsing of elements by presenting in the same view a cluster of items defined as the context. Experiment 2 was designed to compare the Drill-Down method to the Distortion method for locating an object within a pre-specified context. Context is defined as being a set of images spatially and hierarchically related in a certain configuration. As shown in Figure 5, the context for the target (e) is made up of four images (a, b, c and d) and four other files. We hypothesize that participants will locate objects quicker and with fewer errors using the Distortion method over the Drill-down method in a task requiring context.

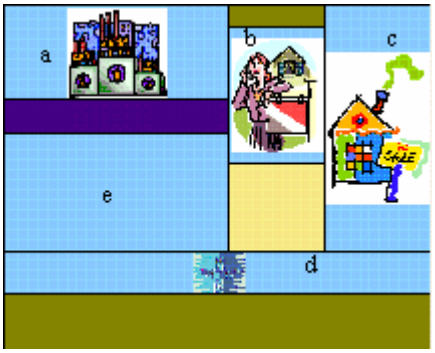

Figure 5: Context of image e (image not shown to participant) consists of images a, b, c, d, and four other files. Image a is a sibling of the target e, image b is in the sibling subtree of the target's parent, and image c is a sibling of the target's parent. Image d is a sibling to the subtree containing a, b, c, and e.

5.1 Method

5.1.1 Subjects

Twelve graduate students participated in the experiment and were randomly assigned to one of the two conditions: Distortion first or Drill-Down first. Subjects had a bachelor degree in either computer science or computer engineering. All were familiar with the concept of file and directory structures and had reasonable experience performing standard file management routines. None had any previous experience using the TreeMap and were not familiar with space-filling concepts.

5.1.2 Materials

As in experiment 1, two different types of hierarchy were used for the experiment: deep and wide. The deep hierarchy was constructed using six levels, with a maximum of three sub-directories per node. The wide hierarchy was created with a depth of three levels, and each node contained a maximum of six sub-directories.

Participants performed the experiment on a 17" monitor with resolution 1024x768 and ran the prototype over Windows XP. The task was described to them before they began the trials.

5.1.3 Procedure

Before starting the experiment, each subject got familiarized with both browsing methods. Once each participant indicated that he or she was comfortable using the tool and its interface, the experiment started.

The task, defined as the context browsing task, consisted of locating a picture within a preconfigured context. The target image was not shown to the user as we wanted the subject to identify the target based on its neighboring images and their interrelations. In this experiment, context is defined by the spatial arrangement and structural relation of objects with respect to the target. Figure 5 is a sample context for image (e) used in the experiment. This task is similar in concept to the sub-structure identification task defined in [8]. By defining such a task, subjects would need to visually maintain the relative positions and relationships between files while browsing for the target image.

Each participant performed the task with three different deep hierarchies and three different wide hierarchies using both methods. Each hierarchy contained seven unique contexts, ten pictures and more than three hundred other types of files. In each task, we randomly chose one context from a hierarchy, and displayed the context to the participant is a separate window outside of the TreeMap.

Two sets of hierarchies (A and B) were used for reducing learning effects. They were constructed with similar structure but different sets of images. Half the participants started the experiment with the Drill-Down method and the other half started using the Distortion method. After completing the tasks in one set of hierarchies with one method, the participants switched onto the other set of hierarchies with the other method. The participant was given the choice to withdraw from the task when he or she could not locate the target picture. For each task the subject was given a maximum time limit of 120 seconds.

We recorded whether the participant located the correct target, whether the participant withdrew from the task, whether the participant exceed the time limit, and the time to execute the task in all cases. In summary, the whole experiment involved: 12 participants × 2 main conditions × 2 types of hierarchies × 3 trials = 144 trials in total.

5.2 Results and Discussion

Context was held constant across all conditions. The time to locate target data was analyzed by means of a 2 × 2 (Type of Method × Hierarchical Structure) univariate analysis of variance (ANOVA), with both Type of Method (Drill Down vs. Distortion) and Hierarchical Structure (3 Nodes Wide vs. 6 Nodes Deep) serving as repeated measures.

An alpha level of .05 was used for all statistical tests. The main effect of Type of Method was found to be significant, $F(1, 140) = 37.87$, $p < .001$, with the Distortion method group's mean task time (38.90 seconds) being faster than the Drill-Down method group's (67.26 seconds). The main effect of Hierarchical Structure was also statistically significant, $F(1, 140) = 5.96$, $p = .02$, with the target being found faster in the Wide Hierarchy (47.45 seconds) than the Deep Hierarchy (58.70 seconds). Finally, an interaction effect was not found between Type of Method and Hierarchical Structure, $F(1, 140) = 1.88$, $p = .17$.

Table 2: Average completion times for Wide and Deep hierarchies with both Methods (standard deviations are in parentheses).

	Wide	Deep
Distortion	36.4 (15.3) sec	41.3 (22.2) sec
Drill-Down	58.5 (32.3) sec	76.1 (35.4) sec

Out of 144 trials, 12 timeouts were recorded over 6 participants. All 12 timeouts were observed when subjects were interacting with the Drill-Down technique. From the 12 timeouts, 10 were reported on the Deep hierarchy. In addition to the 12 timeouts, 3 out of the 144 trials were giveups. All three trials were on the distortion technique. Similarly, only 2 out of 144 trials were recorded as incorrectly found, one on the Distortion and the other on the Drill-Down technique.

These results support our hypothesis in that participants will perform better in a context browsing task with the Distortion method than with the Drill-Down approach. The context browsing tasks assesses the participants' ability of maintaining relations between elements in the structure. In the Distortion approach this

is facilitated as the user, upon opening nodes, can inspect them and decide whether the appropriate relations exist. In the Drill-Down approach participants are required to drill-down and roll-up over several iterations to assess the existence of such relationships. In the Drill-Down approached we observed that in many cases nodes that were previously visited would be visited over again to either confirm their content. We believe this factor to have caused the degradation in performance with the Drill-Down technique.

6. CONCLUSIONS

Interaction is a necessary component for leveraging the power of visualization systems. Without interaction the visualization may only convey partially the information being represented. In this paper we introduce a new interaction technique that facilitates browsing of elements represented in a common space-filling representation known as a TreeMap. In the original design of the TreeMap, data elements are visually acquired by opening successive layers of the hierarchy. A contribution in this paper is a technique that allows users to open nodes and view the contents without opening multiple layers of the hierarchy. This is achieved using distortion and continuous zooming techniques.

Two experiments were designed to evaluate the effectiveness of the new approach. The results of the first experiment suggest that subjects are faster at browsing and locating objects of interest in the distortion technique. The effect is more pronounced when the hierarchy is several layers deep. The main reason that users perform better at the distortion technique is due to the fact that in the drill-down approach the user has to drill-down and roll-up during several iteration until the node is found. As anticipated, the results show that performance with the distortion technique is unaffected by the depth of the hierarchy. Another contribution in this paper is the context browsing task. This task was designed to evaluate the effectiveness of a focus+tcontext technique for locating targets of interest by maintaining interrelations of elements between the focus and context views. In the second experiment, participants were required to locate an object based on its context. The results show that participants were quicker in locating the objects with the Distortion method. These results corroborate with those of [13] in suggesting that interactive techniques employing variations of continuous semantic zooming operations are an enhancement to full zoom approaches (such as drill-down) under certain conditions. It is worth investigating whether such techniques can be applied to current file browsing systems as recent progress has also shown positive results [10].

Additional work for improving the Distortion technique is currently under progress. The distortion algorithm could be modified such that the minimum space taken by distorted nodes is relative to their original size. Another variation of the algorithm could consider distorting more than two nodes at the same time. This could facilitate browsing or performing other tasks such as copying objects from one subtree to another. The distortion technique can also be modified for various other purposes. For instance, an animation based on the distortion could be created on which could be mapped other properties of nodes. The animation rate and distortion amount could be used as two variables in addition to the space and color of a node. Query results on a hierarchy could benefit from such a variation, for instance.

ACKNOWLEDGEMENTS

We are grateful to Richard Zobarich for assisting with the statistical analysis and to Jason Leboe and David Scuse for useful suggestions on drafts of this paper. Thanks to NSERC for supporting this research.

REFERENCES

[1] K. Andrews and H. Heidegger. Information slices: Visualising and Exploring Large Hierarchies using Cascading, Semi-Circular Discs. INFOVIS, pp. 9-12, 1998.

[2] B. B. Bederson. PhotoMesa: A Zoomable Image Browser Using Quantum TreeMaps and Bubblemaps. Proceedings of the ACM Symposium on User Interface Software and Technology, pp. 71-80, 2001.

[3] B. B. Bederson, A. Clamage, M. Czerwinski, G. G. Robertson. DateLens: A fisheye calendar interface for PDAs. ACM Transactions on Computer-Human Interaction 11(1): 90-119, 2004.

[4] B. B. Bederson, James D. Hollan, Ken Perlin, Jonathan Meyer, David Bacon, George W. Furnas. Pad++: A Zoomable Graphical Sketchpad for Exploring Alternate Interface Physics. Journal of Visual Languages and Computing, 7(1): 3-32, 1996.

[5] B. B. Bederson, B. Shneiderman, M. Wattenberg. Ordered and quantum TreeMaps: Making effective use of 2D space to display hierarchies. ACM Transactions on Graphics, 21(4): 833-854, 2002.

[6] Y. Fua, M. O. Ward, E. A. Rundensteiner. Structure-Based Brushes: A Mechanism for Navigating Hierarchically Organized Data and Information Spaces. IEEE Transactions on Visual Computing and Graphics, 6(2): 150-159 (2000)

[7] G. W. Furnas. Generalized Fisheye Views. Proceedings of CHI'86, pp. 16-23, 1986

[8] P. P. Irani, C. Ware. Diagramming Information Structures using 3D Perceptual Primitives. ACM Transactions on Computer-Human Interaction, 10(1): 1-19, 2003.

[9] J. Lamping, and R. Rao. The Hyperbolic Browser: A Focus + Context Technique for Visualizing Large Hierarchies. Journal of Visual Languages and Computing, 7(1): pp. 33-55, 1996.

[10] M. J. McGuffin, G. Davison, R. Balakrishnan: Expand-Ahead: A Space-Filling Strategy for Browsing Trees. INFOVIS, pp. 119-126, 2004.

[11] R. Rao and S. K. Card. Table lens: Merging graphical and symbolic representations in an interactive focus plus context visualization for tabular information. Proceedings of CHI'94, pp. 318-322, 1994.

[12] G. G. Robertson, S. K. Card, J. D. Mackinlay: Information Visualization Using 3D Interactive Animation. Communications of the ACM, 36(4): 56-71, 1993.

[13] D. Schaffer, Z. Zuo, S. Greenberg, L. Bartram, J. Dill, S. Dubs, M. Roseman. Navigating Hierarchically Clustered Networks Through Fisheye and Full-Zoom Methods. ACM Transactions on Computer.-Human Interaction, 3(2): 162-188, 1996.

[14] B. Shneiderman. Tree visualization with TreeMaps: a 2d Space-Filling Approach. ACM Transactions on Graphics, 11(1):92–99, 1990.

[15] J. T. Stasko, E. Zhang. Focus+Context Display and Navigation Techniques for Enhancing Radial, Space-Filling Hierarchy Visualizations. INFOVIS, pp. 57-65, 2000.