Browsing Zoomable Treemaps: Structure-Aware Multi-Scale Navigation Techniques

Renaud Blanch and Éric Lecolinet

Abstract—Treemaps provide an interesting solution for representing hierarchical data. However, most studies have mainly focused on layout algorithms and paid limited attention to the interaction with treemaps. This makes it difficult to explore large data sets and to get access to details, especially to those related to the leaves of the trees. We propose the notion of *zoomable treemaps* (ZTMs), an hybridization between treemaps and zoomable user interfaces that facilitates the navigation in large hierarchical data sets. By providing a consistent set of interaction techniques, ZTMs make it possible for users to browse through very large data sets (e.g., 700,000 nodes dispatched amongst 13 levels). These techniques use the structure of the displayed data to guide the interaction and provide a way to improve interactive navigation in treemaps.

Index Terms—Information visualization, multi-scale interaction, structure-aware navigation, zoomable treemaps.

1 Introduction

Large hierarchical data sets are widespread and many tasks require users to browse such data. For instance, searching for a file in a file system or for a web site in a web directory involves tree navigation. The depth or the breadth of these trees can be quite large: the Open Directory Project (ODP) [1] contains 694,986 web sites categories on 13 levels while the first author's own home directory contains more than 120,000 files on 14 levels.

One major drawback of common ways of representing trees (such as node-link representations) is that they do not use screen real estate very efficiently [23, 26]. More recent techniques, such as treemaps [31] propose an interesting approach to solve this problem. A treemap is a 2D space filling tree visualization that uses most of the available space for displaying leaves. Nodes are represented as nested rectangles and their layout reveals the structure of the tree. Besides, the size of these rectangles can be used to represent a quantitative attribute. Their color or other graphical properties could also be used to represent other attributes. For instance, the size of each node in the representation of the Open Directory Project shown in Figure 1-left indicates the number of links of the corresponding category.

Much attention has been devoted in recent years to enhance the visual aspect of treemaps (e.g., [34, 14, 7, 9]). However, surprisingly, few studies paid attention to the improvement of interaction techniques for navigating treemaps. Yet, treemaps are not very convenient for exploring large data sets, especially when it is necessary to get access to details. The labels of the nodes are likely to be very small, or even illegible, when large trees are displayed using this technique. Hence, efficient interaction techniques are necessary for navigating large treemaps where it is useful to perceive details, such as node labels, or complementary information displayed in the rectangles of the leaves.

Our article focuses on this point by proposing a set of interaction techniques that aim at improving the navigation in treemaps. Because of their recursive nature that makes them inherently multi-scale, treemaps are natural candidates for being used in the same way as zoomable user interfaces (ZUIs) [25, 8]. In the following sections, we propose the concept of *zoomable treemaps* (ZTMs), an hybridization between treemaps and zoomable user interfaces that facilitates the navigation in large hierarchical data sets.

First, we present the proposed interaction techniques for browsing through ZTMs and how these techniques use the structure of the data to improve the navigation. Then we provide details about the implementation of ZTMs, and compare our contribution to related work.

- *Renaud Blanch is with LIG, University of Grenoble 1, E-mail: renaud.blanch@imag.fr.*
- *Éric Lecolinet is with École Nationale Supérieure des Télécommunications (GET), E-mail: eric.lecolinet@enst.fr.*

Manuscript received 31 March 2007; accepted 1 August 2007; posted online 27 October 2007. Published 14 September 2007.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

2 Interaction with Zoomable Treemaps

Browsing very large data sets can make classic treemap interaction techniques quite inefficient. Similarly, simply making treemaps continuously zoomable and using traditional ZUI interaction techniques is not sufficient for browsing data efficiently.

We propose two interaction modes for ZTMs. The first mode is an improvement of traditional treemap interaction techniques: the navigation relies on the ability to make the visualisation focus on nodes that are selected interactively. The second mode is an evolution of ZUIs interaction techniques: it is based on a continuous movement metaphor in the visualisation space. These two modes respectively allow *discrete* and *continuous* interactions. We present them in the next section and we show how they can be seamlessly combined.

2.1 Treemaps Revisited: Discrete Interactions

A reference implementation for treemaps is provided by the Human-Computer Interaction Lab (HCIL) from the Maryland University [27]. The interaction techniques proposed by other publicly available implementations of treemaps (e.g., Discovery [6]) are very similar to the one that is provided by this reference implementation. A node can be selected by clicking on its "title bar" (which must thus reserve enough space to display the node label). Figure 2-left shows an entire tree displayed using the HCIL treemap implementation. Selecting a node changes the view directly, no transition is performed to show the detail of the selected node (Figure 2-right).

The lack of animated transition is not the single problem of this interaction technique. Displaying nodes label in a "title bar" does not scale well for very large and deep trees. Reserving a constant amount of space at each level is quite a space-consuming strategy. As a consequence, the amount of space needed for displaying large trees often exceeds the available space on the screen. A solution to this problem is to avoid using title bars as proposed in [17]. Our zoomable treemap implementation follows the same principle.

This solution has several drawbacks however. First, as they are displayed inside nodes, labels belonging to different layers are often superimposed. This problem strongly reduces the readability of node labels. We will show how this problem can be addressed in the Implementation section. Another problem, that has a direct impact on interaction, is that nodes cannot be directly selected by pointing at them. Without a title bar, each pixel that is displayed for representing

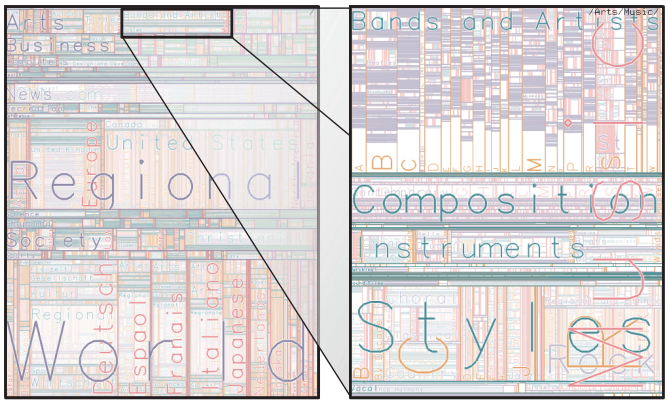

Figure 1. Tree displayed as a zoomable treemap.
(left) 694,986 categories of the ODP [1],
(right) zoomed view providing details on *Arts/Music*.

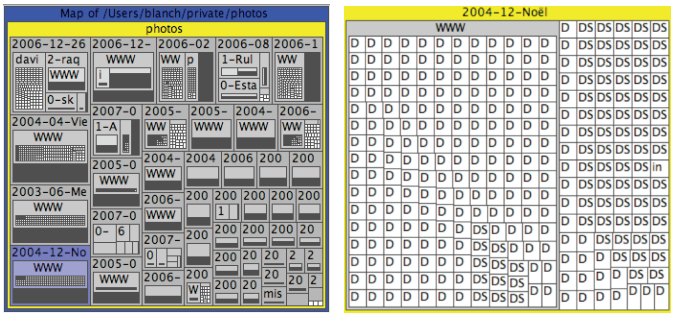

Figure 2. Tree displayed with the HCIL Treemap 4 software.
(left) global view, (right) detail of a node.

a given node also belongs to one of its children (if this node has children). Conversely, because of the recursive construction of treemaps each pixel also belongs to the parent of the node. Thus, a given pixel does not designate a single node in the tree, but a whole branch of nodes (from the root to the smallest leaf node that encloses this pixel). We have taken this problem into account while designing our interaction techniques. They make it possible to navigate in depth as well as in breadth in the tree structure, and to access any visible node directly.

2.1.1 In Depth Navigation

In depth navigation allows a user to go up and down in the tree. As said above, the position of the mouse designates a branch of the tree. A left (resp. right) click drills down (resp. roll up) one level along the branch. In order to go down or up one level, a current node must be defined. This reference node is the smallest node that encloses the whole view. Drilling down selects one child of the reference node, while rolling up selects its parent.

Figure 3[1] illustrates this interaction: on the left, the current node is the root of the tree, which occupies the whole view (black frame). A mouse click (with the left button) launches the interaction that goes down one level from where the pointer is currently located (i.e. to the *People* child of the current node in this example). An animation is then triggered (Figure 3-center), that makes the targeted node fill the whole window (Figure 3-right), and thus become the new reference node. This animation is performed in such a way that each step involves a constant scale factor and a constant apparent translation. It aims at helping the user to keep her spatial orientation.

[1] The data set used in the Figures 3–8 is the one that was used for the *Great CHI'97 Browse Off* [24].

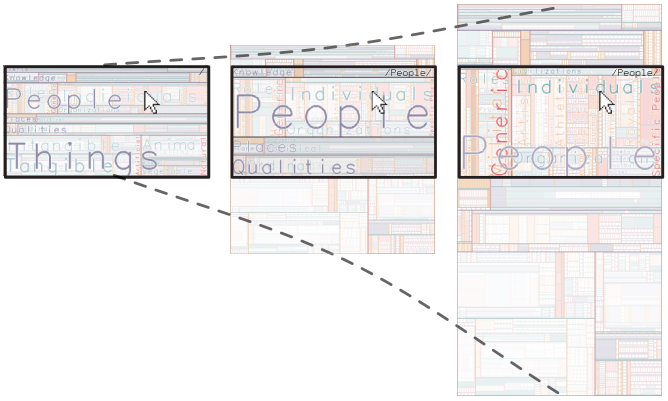

Figure 3. In depth navigation.
Drilling down one level from the root to the *People* category.

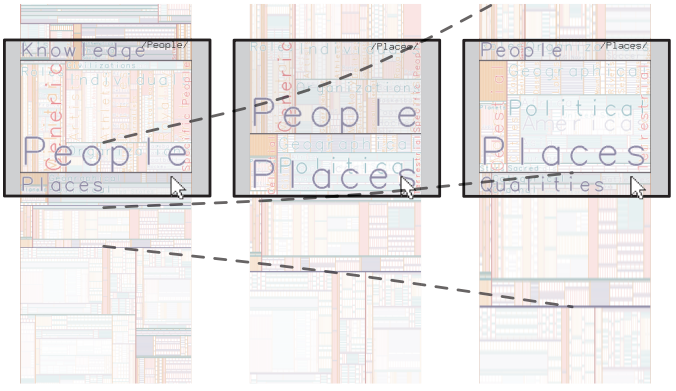

Figure 4. In breadth navigation.
A click in the margin "flips" to the next node.

2.1.2 In Breadth Navigation

It is also often desirable to navigate *in breadth*, from a node to its siblings. For example, users may want to browse through the files contained in a directory without going back to their common ancestor. In depth navigation is not appropriate in such a case.

We provide *in breadth* navigation with the *flipping* interaction. This interaction allows to switch quickly from a node to one of its neighbours. To enable this interaction, the view is surrounded with a frame that displays the neighbourhood of the current node (Figure 4). Clicking in this frame (Figure 4-left) triggers an animation (Figure 4-center) that centers the view on the sibling of the previously current node (Figure 4-right). This animation is fast enough to be rapidly repeated, making the user feel like flipping the pages of a book.

2.1.3 Direct Node Selection

The techniques presented so far only make it possible to select nodes that are on the same layer as the current node, or located one level above or below. This limitation is a consequence of the fact that a click cannot designate a specific node in a treemap: each point belongs to a single leaf node but also to all its ancestors. However, since treemaps display several layers of the tree simultaneously and provide visual access to them, a technique to select any visible node is most desirable. As a solution, we propose *direct access* node selection by means of gesture interaction.

Direct node selection permits to select a node by considering a stroke rather than a position. The selected target is then the smallest node containing the stroke (Figure 5). When the user starts a stroke, the smallest node that encloses it is necessarily a leaf of the tree (Figure 5-1). Then, the cursor can cross a boundary of the current target

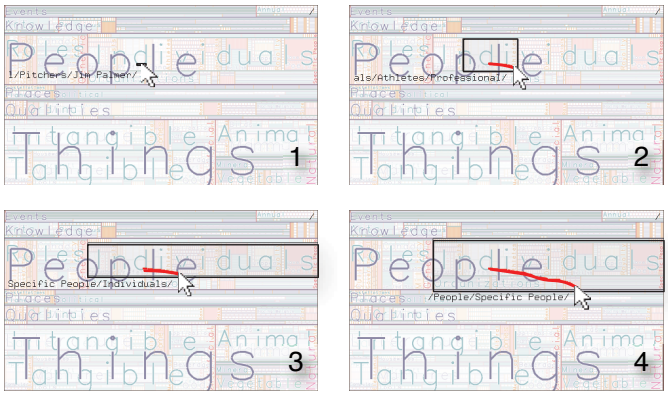

Figure 5. Direct acces navigation.
A crossing-based interaction enables direct access to visible nodes.

during the drawing gesture. The target is then changed to the smallest ancestor node that contains the entire stroke (Figures 5-2, 5-3 and 5-4). Visual feedback is provided dynamically to display the current target. Finally, an animation that makes the target fill the view is triggered when the mouse button is released.

This technique can be seen as crossing-based interaction [3], a kind of technique that has been recently proposed as an alternative to classical activation [4, 16]. Using our technique, it is possible to traverse an arbitrary number of tree layers for accessing a specific node, just by crossing node borders interactively.

Moreover, the surrounding frame used for the flipping interaction (that was previously described) finds a second usage here: by making a stroke starting in the view and ending outside, a border of the node that fills the view is necessarily crossed. As a consequence its parent is selected by drawing this stroke. Consequently, the crossing interaction makes it also possible to roll up in the tree by using a gesture that is consistent with the other interaction techniques.

2.2 ZUIs Revisited: Continuous Interaction

The previous interaction techniques are discrete in the sense that they permit the selection of a given node. However, they have a common limitation: only nodes that are large enough to be displayed when their parents occupy the whole view can be selected. In large and unbalanced trees, or when the window is small, nodes are often very small compared to their parents. In this case, using a navigation technique that does not rely on node selection is essential.

We introduce *zoomable treemaps* to overcome this limitation. Since treemaps are inherently multi-scale, we propose to consider them as zoomable spaces. However, the pan-and-zoom navigation of ZUIs [25, 8] is not well adapted for treemaps. Since treemap layout algorithms can produce rectangles with variable aspect-ratios, pure geometric zoom leads to views where the elongated nodes can only partially fit. Figure 6-left shows the effect of a pure geometric zoom: the rectangles are so thin that it is impossible to display a label inside them and, as a consequence, users get rapidly lost.

2.2.1 Snap-Zoom

We introduce *snap-zoom*, an interaction that magnifies the horizontal and vertical axes by using different scales so that the aspect ratio of the target node is gradually changed to finally match the window (Figure 6-right). The scale of the view is controlled by the mouse wheel, and the zoom effect is centered on the position of the pointer. The resulting effect is a smooth zoom which regularly expands or shrinks the treemap while deforming it so that the layers that are traversed successively fit the window one after another. Hence the name *snap-zoom*.

An interesting aspect is that the actual distribution of the scale between the horizontal and vertical dimensions is determined by the system. The details of this computation are given in the Implementation section. The scaling uniformly affects the treemap, so that the amount of information that is mapped on the nodes is always relevant: nodes

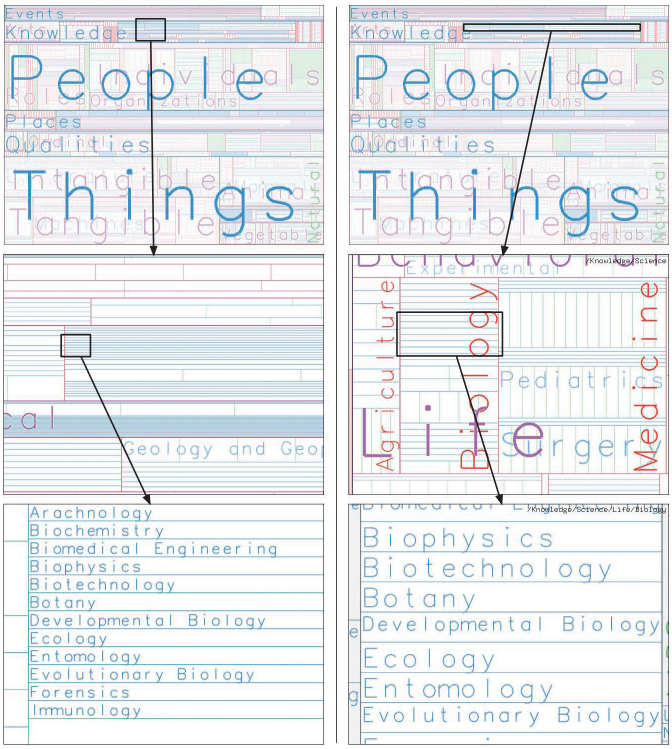

Figure 6. Snap-zoom. (left) geometric zoom, (right) snap-zoom.

can thus be visually compared. This contrasts with techniques such as Fisheye views [18], that do not have this property.

Although the distribution of the scale between the two dimensions is determined by the system, the control feels natural because the user manipulates them through a single dimension that has a physical meaning: the global scale of the document.

2.3 Multiplexing Discrete and Continuous Interaction

The animations used by the discrete interaction technique also change the scales of the *x*- and *y*-axis independently. The discrete and continuous interaction techniques have a consistent behaviour and can be mixed seamlessly: snap-zoom and drilling down interactions use the same transitions between successive layers. But for the continuous interaction techniques, the control of the animation is mapped onto the scale dimension and controlled by the user as in the StyleCam interaction technique [15].

This homogeneity in performing animations supports the harmonious combination of discrete and continuous interaction. The *zoom menu* illustrates this capability.

2.3.1 Zoom Menu

As mentioned before, a given point on the display identifies a branch of nodes. The *zoom menu* is a contextual menu that appears when the user presses the mouse and that shows the labels of the branch nodes corresponding to the mouse press location. This menu is positioned in such a way that the cursor appears on the top of the current node (which is the node that currently occupies the whole window space, as the *Life* category in Figure 7-a). The labels of its ancestors are located above this node in the menu (e.g., *Science, Knowledge* and *Categories,* the tree root in this example). Conversely, the labels of the direct and indirect children in the branch (e.g., *Biology* and *Biochemistry*) appear below the current node in the zoom menu. The user can select any node in the branch by dragging the mouse and releasing it on the desired label. The corresponding node is then selected and becomes the new current node. An animation is then performed. It continuously increases the size of this node until it occupies the whole window space (Figures 7-c and 7-d).

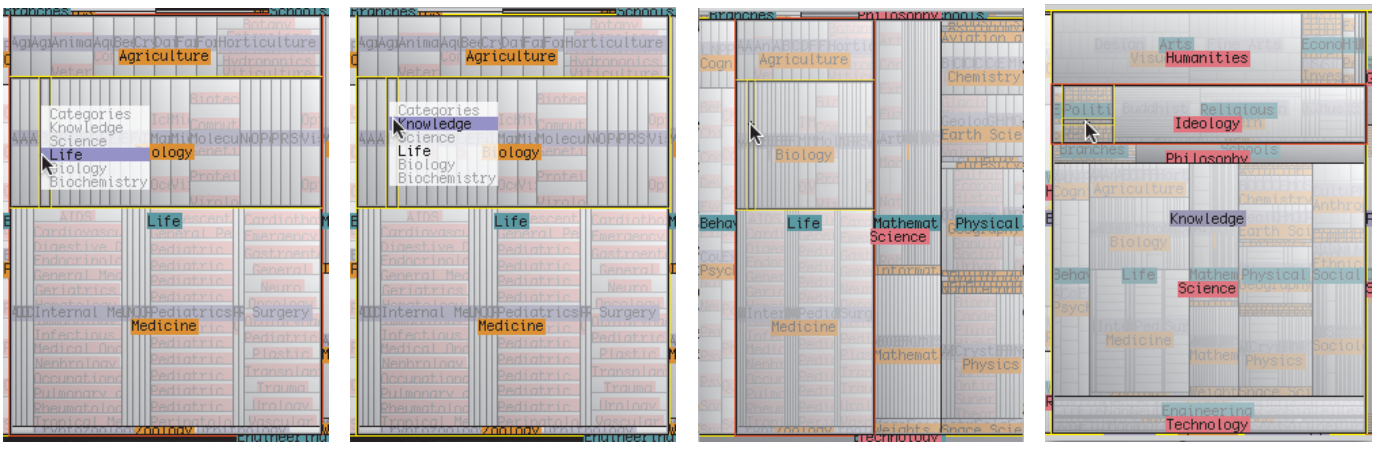

Figure 7. Zoom menu, discret mode: (a, b) target selection, then (c, d) animation.

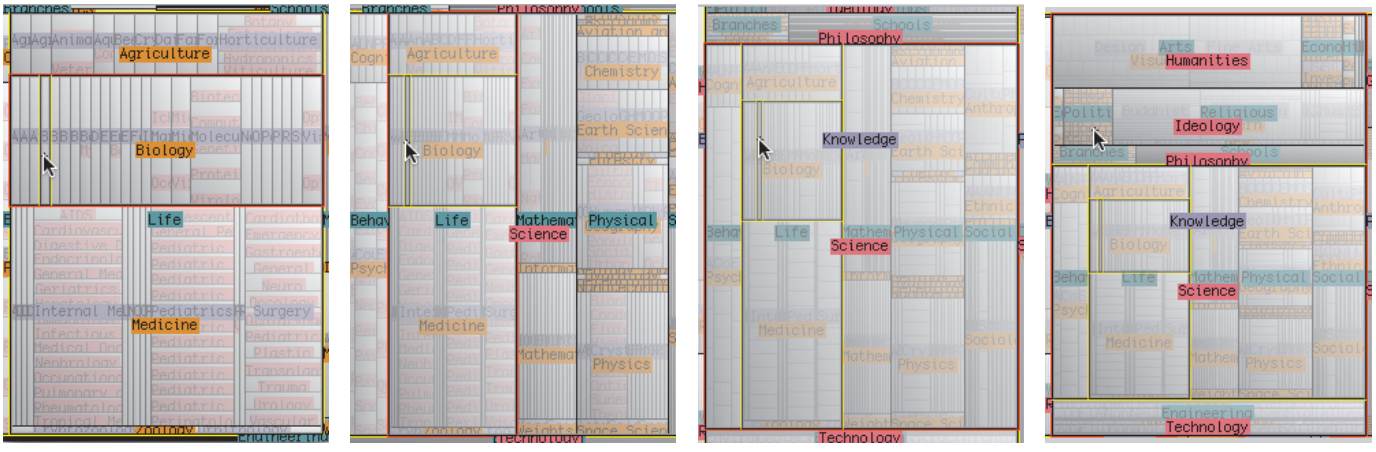

Figure 8. Zoom menu, continuous mode: the interaction controls the animation directly.

In addition to this discrete interaction technique, the zoom menu also permits continuous interaction. In this case, the zooming level of the document is continuously controlled by dragging the mouse along the y-axis (a zoom-out effect is obtained by moving the mouse upwards in the window, and vice versa). This effect is shown in Figure 8 where the user moves up in the tree structure by dragging the mouse upwards in the window.

These two different ways of interacting (through a discrete or continuous interaction) are always both available. They are triggered in the same way as the novice and expert modes in Marking menus [22]. If the user presses the mouse and waits for a small delay (about 0.3 ms), the Zoom menu appears and the discrete mode is enabled. However, the user can also drag the mouse immediately, without waiting for the menu to appear, in which case the continuous mode is triggered (as in expert mode, the menu will not appear in this case). This interaction technique can also be seen as an improvement of Control menus [28], an extension of Marking menus that makes it possible to select and control continuous interactions in a single gesture.

3 Implementation of Zoomable Treemaps

The software prototype developed to test and design our interaction techniques was written in C++. OpenGL was used for rendering and GLUT for font and window management. This section will now detail some interesting implementation aspects. We first present the algorithms our interaction techniques rely on. Techniques for making text rendering legible are then explained. Finally, we describe several optimizations that make it possible to preserve interaction fluidity even when browsing very large data sets.

3.1 Interactions

Two problems had to be solved to combine the interaction techniques presented so far in a coherent way. First, discrete interactions are based on the notion of a current node that occupies the whole window space. However, continuous interactions can leave the system in a state where there is no actual current node. This notion must thus be generalized in order to precisely define the reference node that is used by all interaction techniques in all situations. Another issue concerns the effect of zoom level changes on the horizontal and vertical layout of displayed data. In fact, nodes are not necessarily zoomed in the same way along the x- and y-axis in order to optimize their rendering.

3.1.1 Current node selection

The current node can be defined as the largest node that is currently shown on the window. As seen before, a point on the screen identifies a branch of the tree that is currently displayed. Hence, by taking into account the location of the mouse, the current node can be found by searching along the branch that corresponds to this location. Following treemaps construction, it is always possible to find a couple of nodes (p,c) so that c is a child of p, c is completely included in the window space and p exceeds this area. These two nodes are natural candidates for defining the "current node".

Depending on the interaction type and the direction they are performing on, p or c are either chosen as the current node. For instance, interaction techniques that perform in depth navigation use c if the user is zooming in, and p if he is zooming out. By convention, p is always used as the reference node when other interaction techniques are used.

3.1.2 Zooming algorithm

Once the current node is determined, the appropriate zoom factor must be applied in the x- and y-directions. A predefined zooming step is used when animations are performed, while the zoom factor depends on mouse movement for user interactions. The surfacic zoom factor δs^2, must satisfy the following constraints: $\delta s^2 > 1$ when zooming in and $\delta s^2 < 1$ when zooming out. This global zoom factor must then be distributed among x- and y-axis in such a way that the new current node will have the appropriate aspect ratio to fill the whole window space. These horizontal and vertical scale factors, δs_x and δs_y, must respect the constraint $\delta s^2 = \delta s_x \times \delta s_y$ in order to obtain a valid zooming effect. For this purpose, the global zoom factors for the x- and the y-axis (Δs_x and Δs_y) are first computed in such a way that the current node would fill the window:

$$\Delta s_x = W/w, \;\; \Delta s_y = H/h$$

with W, H being the width and height of the window and w, h the initial width and height of the current node.

The scale factor δs^2 is then distributed according to the relative weight of the two components of the global magnification factor $\Delta s^2 = \Delta s_x \times \Delta s_y$. However, as constant speed zooming requires a geometric progression of scale, the proportions must be considered for the zoom index, that is to say the logarithm of the scale [20]. With $z = \log(s)$ taken as the zoom index, the following formulas are obtained:

$$\delta z_x = \frac{\delta z^2}{\Delta z^2} \Delta z_x, \;\; \delta z_y = \frac{\delta z^2}{\Delta z^2} \Delta z_y \text{ and finally:}$$

$$\delta s_x = {\Delta s_x}^t, \;\; \delta s_y = {\Delta s_y}^t, \text{ with } t = \frac{\log \delta s^2}{\log \Delta s^2}.$$

3.2 Rendering

The visualization of large information sets requires minimization of the amount of unused space while avoiding visual overload. In order to solve this difficult but crucial problem, node rendering is performed by starting with the innermost nodes, so that higher levels are not occluded by the details contained in lower layers. Four colors are used in turn, each layer of the hierarchy being rendered using one of them. These colors belong to a color set designed in such a way that they can be easily differentiated by human readers [13]. This strategy makes it possible to improve the readability of node labels contained in higher layers even if they are superimposed on labels contained in underlying layers. Readability is also reinforced by surrounding letters with a thin white outline.

A fading effect is used to make labels located in lower levels less visible. This effect is controlled in such a way that details get progressively revealed while in depth navigation is being performed. Similarly, higher levels that are located on the top of the current node are not drawn in order to avoid visual occlusion. For this purpose, layers are distributed among a virtual z-axis that is perpendicular to the screen space. The rendered view moves into this 3D space according to the zooming level. This makes it possible to discard upper layers automatically thanks to OpenGL capabilities. Since an orthographic projection is used, the 2D positions of the nodes are not affected by the variations along the z-axis.

3.3 Optimizations

Three types of optimizations are done in order to obtain frame rates that are compatible with interactive use (that is to say, a minimum rate of 10 fps). First, a semantic zoom is used: labels that would not be readable because of a exceedingly small size are not rendered. This simple optimization is quite efficient because text rendering is a rather costly operation. Other optimizations are related to node characteristics. Nodes that are located outside the window are clipped for obvious reasons. Following treemaps construction principles, their children can also be discarded. Entire branches can thus be clipped just by traveling through the tree from its root node. Nodes whose size is less than a given threshold (e.g., one pixel) are also ignored. This threshold is dynamically adapted according to the current real time performance of the system, so that more details are displayed when there is enough time to improve rendering without compromising interaction fluidity. This property makes it possible to use our system on a variety of hardware configurations, even those that only provide modest capabilities.

For example, on Figure 1-left, the tree consists of 697,986 nodes but only 145,841 are bigger than one pixel and displayed, and only 235 labels are shown.

3.4 Prototype

The interaction techniques presented so far have been implemented in a prototype that is freely available[2]. A preliminary version was already presented as a demonstration [12].

4 Related Work

4.1 Zoomable Treemaps

As said in the introduction, a reference implementation of treemaps is provided by the HCI Lab at the University of Maryland [27]. It provides a single navigation technique: double clicking on a node or a subtree border zooms in and a click with the right mouse button zooms out. Other available treemap implementations (see [32] for a comprehensive list) are equivalent in terms of interaction. They do not focus on the interaction issues, probably because visualization of trees is a challenge by itself. PhotoMesa [7] uses treemaps to display and browse images in a file system. It allows to zoom continuously, but the underlying hierarchical structure of the file system is collapsed onto a flattened view. StepTree [11] uses a continuous zoom to animate the transitions when the user navigates between levels of the treemap, but it does not provide her with an interactive control of the scale. Matrix Zoom [2] provides a *smooth pan-and-zoom* interaction based on work by van Wijk and Nuij [33]. This kind of animation would be useful in our system, but this technique seems hard to adapt to an interaction that is continuously controlled by the user.

Zoomable interfaces, that were introduced by Perlin and Fox [25] propose several interaction techniques: portals, panning and zooming. With zoomable treemaps, we have relaxed the constraint of geometric magnification. By allowing different scales on the two axes, our techniques are adapted to any rectangular treemap layout algorithm: *slice-and-dice* [31], *squarified* [14], or *ordered* [9]. Crossing-based selection and snap-zoom are improvements over classical techniques because they can pass through several layers of the multi-scale tree in a single interaction. They could be as well adapted to non-rectangular layouts such as *voronoi treemaps* [5] or even *generalized treemaps* [35] since they only rely on the crossing of borders. Shi et al. [30] have evaluated a navigation technique where the treemap is deformed by using a fisheye technique, hence making it possible to traverse several layers. However, it does not preserve the layout of the treemap. *Elastic hierarchies* [36] combine treemaps and node-link diagrams to visualize trees. To solve the problem of node selection Zhao et al. added a tab outside the treemap on which the user can select the level of detail at which she wants to interact.

4.2 Dealing with Large Trees

Large trees have already been displayed using treemaps [17], but this work was mostly focused on visualization, rather than interaction. SpaceTree [26] and *revisited degree-of-interest trees* [21] are some of the recent contributions to the interaction with node-link representations. They both try to effectively use the screen real estate while providing insights about the non-visible parts of the tree. Displayed nodes are chosen according to various policies that may depend on explicit user clicks, textual queries, or a computed degree of interest. The use of animations to help the user to understand how the layout changes has been suggested for treemaps by Ghoniem and Fekete [19] and sophisticated tree animations have been proposed by Plaisant et al. [26].

Advanced interaction techniques using animation have been proposed for nested representations of trees. With *structural zooming* [29]

[2] The source code is available at: http://iihm.imag.fr/blanch/projects/ztm/.

the user can reveal or hide children of nodes by clicking on them. This triggers the computation of a new layout that shows the desired nodes, and an animation is performed so that nodes smoothly reach this layout. McGuffin et al. [23] propose an algorithm to *expand-ahead* the tree while the user is navigating inside. Their algorithm computes a layout that shows a maximum of presumably interesting nodes for a given user focus. An animation is also used to reach this layout.

5 Conclusion & Future Directions

We have introduced the concept of zoomable treemaps and a consistent set of interactions for browsing them. They make it possible to explore very large trees displayed as zoomable treemaps. These interaction techniques use the structure of the tree to guide the navigation and reduce the chance for the user to get lost in the information space. They do not constrain unnecessarily the user but only assist her free navigation. They integrate seamlessly discrete and continuous control. Our first usability tests gave positive results and very encouraging qualitative feedback. We now plan to conduct a controlled evaluation to compare our interaction techniques with other tree browsing techniques.

We also want to adapt those techniques for a broader class of devices and of interaction contexts. For example, handheld devices have very limited screen size and require pen based interaction, and our zoom menu could be very well adapted for such configurations. On the other hand, table-top interaction allows collaborative interaction and large display surfaces (like Bérard's Magic Table [10]) which may open a new design space for the interaction with very large data sets.

Acknowledgements

The authors wish to thank the anonymous reviewers for their useful comments and suggestions about this paper. We also would like to thank J. Coutaz, S. Dupuy-Chessa, and G. Godet-Bar for their help in preparing the final version of this paper.

This work was supported in part by a grant from the Regional Council of Île-de-France.

References

[1] Open directory project, 2006. http://www.dmoz.org.

[2] J. Abello and F. van Ham. Matrix Zoom: A visual interface to semi-external graphs. In *Proc. IEEE InfoVis'04*, pages 183–190, Washington, DC, USA, 2004. IEEE Computer Society.

[3] J. Accot and S. Zhai. More than dotting the i's — foundations for crossing-based interfaces. In *Proc. ACM CHI'02*, pages 73–80, 2002.

[4] G. Apitz and F. Guimbretière. CrossY: a crossing-based drawing application. In *Proc. ACM UIST'04*, pages 3–12, 2004.

[5] M. Balzer and O. Deussen. Voronoi treemaps. In *Proc. IEEE InfoVis'05*, page 7. IEEE Computer Society, 2005.

[6] T. Baudel. From information visualization to direct manipulation: extending a generic visualization framework for the interactive editing of large datasets. In *Proc. ACM UIST'06*, pages 67–76, 2006.

[7] B. B. Bederson. PhotoMesa: a zoomable image browser using quantum treemaps and bubblemaps. In *Proc. ACM UIST'01*, pages 71–80, 2001.

[8] B. B. Bederson and J. D. Hollan. Pad++: a zooming graphical interface for exploring alternate interface physics. In *Proc. ACM UIST'94*, pages 17–26, 1994.

[9] B. B. Bederson, B. Shneiderman, and M. Wattenberg. Ordered and quantum treemaps: Making effective use of 2d space to display hierarchies. *ACM Trans. Graph.*, 21(4):833–854, 2002.

[10] F. Bérard. The magic table: Computer-vision based augmentation of a whiteboard for creative meetings. In *Proc IEEE PROCAM'03*, 2003.

[11] T. Bladh, D. A. Carr, and M. Kljun. The effect of animated transitions on user navigation in 3D tree-maps. In *Proc. IEEE IV'05*, pages 297–305, 2005.

[12] R. Blanch and É. Lecolinet. Navigation techniques for zoomable treemaps. In *Adj. Proc.: Demos of ACM UIST'06*, pages 49–50, 2006.

[13] C. A. Brewer. ColorBrewer - selecting good color schemes for maps, 2006. http://www.ColorBrewer.org.

[14] M. Bruls, K. Huizing, and J. J. van Wijk. Squarified treemaps. In *Proc. Eurographics & IEEE TCVG Symp. on Visualization*, pages 33–42, 2000.

[15] N. Burtnyk, A. Khan, G. W. Fitzmaurice, R. Balakrishnan, and G. Kurtenbach. StyleCam: Interactive stylized 3D navigation using integrated spatial and temporal controls. In *Proc. ACM UIST'02*, pages 101–110, 2002.

[16] P. Dragicevic. Combining crossing-based and paper-based interaction paradigms for dragging and dropping between overlapping windows. In *Proc. ACM UIST'04*, pages 193–196, 2004.

[17] J.-D. Fekete and C. Plaisant. Interactive information visualization of a million items. In *Proc. IEEE InfoVis'02*, pages 117–123, 2002.

[18] G. W. Furnas. Generalized fisheye views. In *Proc. ACM CHI'86*, pages 16–23, 1986.

[19] M. Ghoniem and J.-D. Fekete. Animating treemaps. In *Proc. Workshop on Treemap Implementation and Applications*. University of Maryland, 2001.

[20] Y. Guiard and M. Beaudouin-Lafon. Target acquisition in multiscale electronic worlds. *Int. J. Human-Computer Studies*, 61:875–905, 2004.

[21] J. Heer and S. K. Card. DOITrees revisited: Scalable, space-constrained visualization of hierarchical data. In *Proc. AVI'04*, pages 421–424, 2004.

[22] G. P. Kurtenbach. *The design and evaluation of marking menus*. PhD thesis, University of Toronto, 1993.

[23] M. J. McGuffin, G. Davison, and R. Balakrishnan. Expand-ahead: a space-filling strategy for browsing trees. In *Proc. IEEE InfoVis'04*, pages 119–126, 2004.

[24] K. Mullet, C. Fry, and D. Schiano. On your marks, get set, browse! (the great CHI'97 browse off), 1997. http://www.sigchi.org/chi97/proceedings/panel/kem.htm.

[25] K. Perlin and D. Fox. Pad: an alternative approach to the computer interface. In *Proc. ACM SIGGRAPH'93*, pages 57–64, 1993.

[26] C. Plaisant, J. Grosjean, and B. B. Bederson. SpaceTree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In *Proc. IEEE InfoVis'02*, pages 57–64, 2002.

[27] C. Plaisant and B. Shneiderman. Treemap: Home page, 2006. http://www.cs.umd.edu/hcil/treemap/.

[28] S. Pook, É. Lecolinet, G. Vaysseix, and E. Barillot. Control menus: execution and control in a single interactor. In *Ext. abs. CHI'00*, pages 263–264, 2000.

[29] K. Pulo, P. Eades, and M. Takatsuko. Smooth structural zooming of h-v inclusion tree layouts. In *Proc. CMV'03*, 2003.

[30] K. Shi, P. Irani, and B. Li. An evaluation of content browsing techniques for hierarchical space-filling visualizations. In *Proc. IEEE InfoVis'05*, pages 81–88, 2005.

[31] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11(1):92–99, 1992.

[32] B. Shneiderman. Treemaps for space-constrained visualization of hierarchies, 2006. http://www.cs.umd.edu/hcil/treemap-history/.

[33] J. J. van Wijk and W. A. A. Nuij. A model for smooth viewing and navigation of large 2D information spaces. *IEEE Trans. Vis. Comput. Graph.*, 10(4):447–458, 2004.

[34] J. J. van Wijk and H. van de Wetering. Cushion treemaps: Visualization of hierarchical information. In *Proc. IEEE InfoVis'99*, pages 73–78, 1999.

[35] R. Vliegen, J. J. van Wijk, and E.-J. van der Linden. Visualizing business data with generalized treemaps. *IEEE Trans. Vis. Comput. Graph.*, 12(5):789–796, 2006.

[36] S. Zhao, M. J. McGuffin, and M. H. Chignell. Elastic hierarchies: combining treemaps and node-link diagrams. In *Proc. IEEE InfoVis'05*, pages 57–64, 2005.