HeiankyoView: Orthogonal Representation of Large-scale Hierarchical Data

Takayuki ITOH, Koji KOYAMADA

Academic Center for Computing and Media Studies, Kyoto University

itot@computer.org, koyamada@kudpc.kyoto-u.ac.jp

Abstract. Visualization is very useful for various large-scale computing fields. One of the authors has reported a hierarchical data visualization technique, and applied it to various large-scale computing data including large-scale Web sites, but also for data mining results, Web search queries, network intrusion detection results, and distributed processes. This paper presents the new hierarchical data visualization technique, HeiankyoView, which can be also applied to various large-scale computing fields. It places data items of input data while it packs the items as many as possible in small display areas, and aligns the items along X-axis and Y-axis of the display area. The technique represents nodes of hierarchical data as a set of rectangles, which are parallel to X-axis and Y-axis. Here the technique orthogonally divides the display area by extension lines of edges of previously placed rectangles. By referring the grid-like subspaces of the display area, the technique quickly finds the adequate positions to place remaining rectangles.

1. Introduction

There are so many kinds of hierarchical data in our daily life, such as file systems of computers, human resources organization of large companies, and category-based Web sites. Visualization of hierarchical data is therefore a very active research topic. Some of the visualization techniques first represent higher-level portions of the hierarchical data, and provide user interfaces to interactively explore lower-level portions. Others represent the overview of the data by placing all portions into display spaces.

One of the author proposed Data Jewelry Box [Ito03][Yam03], which represents large-scale hierarchical data as a set of nested rectangles. The technique represents leaf-nodes as colored icons, and non-leaf-nodes as rectangular borders.

Figure 1 shows an example of visualization of a Web site using Data Jewelry Box. Forming thousands of Web pages as hierarchical data according to directory structure, the technique places all icons denoting the Web pages onto one display space. As shown in this example, the technique is suitable for representation of large number of leaf-nodes in one display space. The technique has been applied to various large-scale computing fields, not only for large-scale Web sites, but also for data mining results, Web search queries, network intrusion detection results, and distributed process monitoring [Yam03].

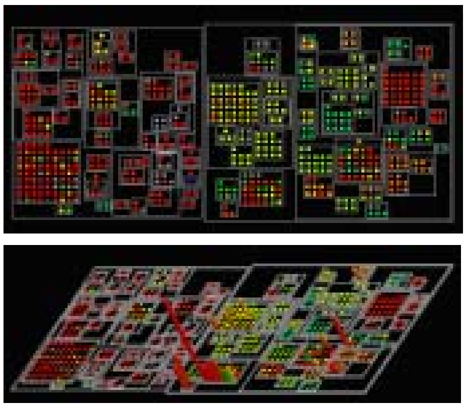

Figure 1. Examples of the representation of hierarchical data. Leaf-nodes of this data denote the Web pages of a Web site, and the non-leaf-nodes of the data denote the directory hierarchy. The leaf-nodes are colored according to their last update time, and given heights according to their access frequencies.

This paper presents the new hierarchical data visualization technique, HeiankyoView. It represents hierarchical data as a set of nested rectangles, similar to Data Jewelry Box. It places data items of input data while it packs the items as many as possible in small display areas, and it orthogonally aligns the items. Conditions for placing the data items are quite similar to those of Data Jewelry Box; however, algorithm of HeiankyoView is quite different from Data Jewelry Box, and therefore visualization results of HeiankyoView are also much different from ones of Data Jewelry Box.

The name of the proposed technique comes from Heiankyo, an ancient palace in Japan. It is well-known that land arrangement of Heiankyo was very well-organized by grid-like blocking. Since our orthogonal representation looks like the land arrangement of Heiankyo, we named our technique HeiankyoView.

2. Related Works

This section provides a brief survey of existing hierarchical data visualization techniques.

2.1 Tree-based visualization

Tree representation is the most popular hierarchical data visualization technique, used in many programs such as well-known file system viewers. Several variations, such as the Hyperbolic Tree [Lam96], Cone Tree [Car95], Fractal Views [Koi95], have been described for the interactive visualization of large-scale tree data sets. These techniques are suitable for visualizing the higher-level data first, and then exploring the lower-level data according to users' choices. They are also suitable for visualizing the connectivity among nodes. Some of such works tried to display thousands of nodes, but results of these studies do not always efficiently use display spaces, because they often occur sparse layouts.

2.2 Space-filling visualization

A Treemap [Joh01], which represents hierarchical data in a manner like nested column charts, is a well-known space-filling approach for hierarchical data visualization. The space-filling approach is especially useful for representations appropriate to visualize quantitative data attributes, as opposed to seeing connectivity, for which tree-based techniques may be more appropriate. This approach is also suitable for representing all of the data in a compact display space. The technique presented in this paper is close to Treemap in terms of providing overviews of the data.

Recently several improved Treemaps techniques have been proposed. Quantum treemap [Bed02] is a very close work to our technique because it is suitable for the representation of categorized data items as icons. As described in [Bed02], the stable layout of dynamic data is a common future research direction of Treemaps.

2.3 Semitransparent 3D visualization

Information cubes [Rek93] is a 3D hierarchical data visualization approach, which positions data in nested semitransparent hexahedrons. H-BLOBS [Spr00] is also a 3D hierarchical data visualization approach, positioning data in nested semitransparent isosurfaces. These approaches have limitations in that they require a graphics environment supporting 3D semitransparent rendering, and in that users' need 3D manipulation skills. Also, the computation times for data layout are not clearly mentioned in their papers.

2.4 Data Jewelry Box

Data Jewelry Box [Ito03] represents hierarchical data as nested rectangles, as shown in Figure 1. It places data items starting from the lowest level of the hierarchical data, and repeats the placement towards the top level. HeiankyoView applies the same representation and processing flow as described in Section 3.

To efficiently place the data items, Data Jewelry Box searches for gaps to place the data items by referring Delaunay triangular meshes connecting centers of previously placed data items.

Recently it has been extended for the stable representation of time-varying hierarchical data [Yam03]. The extension is one of the main advantages of Data Jewelry Box against other existing hierarchical data visualization techniques. The extension refers the previous display layout results, and places the data items so that these positions are enough close to the previous positions.

3. Hierarchical Data Visualization by Nested Rectangles

Similar to Data Jewelry Box, HeiankyoView also represents leaf-nodes as colored icons, and non-leaf-nodes as rectangular borders. HeiankyoView first tightly packs icons in rectangular regions, and then repeats the packing of rectangular regions to represent the hierarchy.

Figure 2 shows an illustration of the order of hierarchical data layout of HeiankyoView. The technique first packs icons (painted square dots in Figure 2) that denote leaf nodes, and then encloses them by rectangular borders that denote non-leaf nodes. Similarly, it packs a set of rectangles that belong to higher levels, and generates the larger rectangles that enclose them. Repeating the process from the lowest level toward the highest level, the technique places all of the data items onto the layout area.

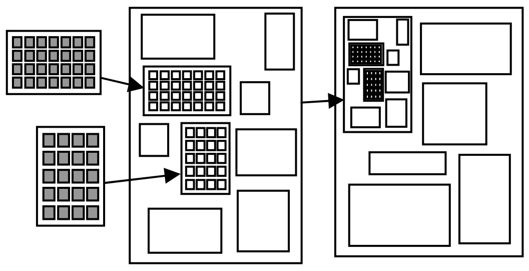

Figure 2. Processing flow of HeiankyoView. It first places data items in the lowest level, and then repeats the placement towards the top level.

Treating icons, thumbnails, and rectangular borders as a set of rectangles, HeiankyoView places the rectangles tightly inside a small rectangular area, and encloses them by a bigger rectangular border that denotes the parent non-leaf node.

If all nodes in a level are leaf-nodes and their sizes and aspect ratios are not specified, our implementation assumes that the all nodes are represented by equal squares. In this case the implementation places them onto an orthogonal regular grid, without using the rectangle packing algorithm. When the level contains n_l nodes, the implementation calculates the rounded value of $\sqrt{n_l}$, and sets the horizontal or vertical number of squares to that value.

If the sizes and aspect ratios of the rectangles in a level vary, the technique needs more robust rectangle packing algorithms. Sections 4 present the detail of rectangle packing algorithm applied to HeiankyoView.

4. Rectangle Layout by Grid-like Subdivision of Display Spaces

4.1 Overview

As mentioned in Section 3, HeiankyoView treats icons, thumbnails, and rectangular borders as a set of rectangles. Given the set of rectangles under a non-leaf-node, HeiankyoView places them onto the display space one-by-one. Here, it places the rectangles while it satisfies the following two conditions:

[Condition 1] It places rectangles so that they do not overlap each other.
[Condition 2] It tries to place rectangles where it minimizes the display area after the placement of each rectangle.

This section describes the algorithm for placing rectangles. Pseudo-code of the presented algorithm is shown in Figure 7. Remark that the algorithm does not entirely minimize the display areas, but just finds configurations that archive local minimum display areas. We think that such local minimum configurations are sufficient for the purpose of HeiankyoView.

4.2 Grid-like subdivision of display spaces

Suppose that several rectangles have been already placed onto a display space. HeiankyoView maintains the occupancy of rectangles in the display space by grid-like subdivision, as shown in Figure 3. It divides the display space by extension lines of edges of previously placed rectangles. It maintains if each rectangular subspace is already filled by placed rectangles or not. Figure 3(right) denotes that painted subspaces have been already filled by placed rectangles, and others are not filled.

Supposing that the display space is divided to p layers along x-axis, and q layers along y-axis, this paper denotes the sorted x-coordinates of the extension lines as x_0 to x_p, and the sorted y-coordinates of the extension lines as y_0 to y_q. HeiankyoView traverses $p \times q$ subspaces to find adequate positions to place remaining rectangles.

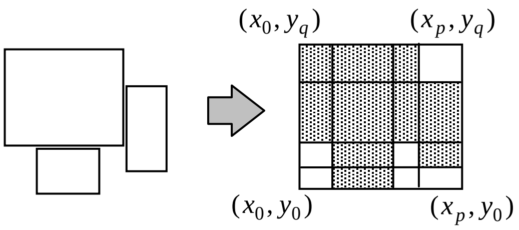

Figure 3. (left) Already placed rectangles. (right) Grid-like subdivision of a display space by extension lines of edges of already placed rectangles.

4.3 Traverse of candidate positions to place rectangles

To decide a position of remaining rectangles, HeiankyoView evaluates multiple candidate positions while it traverses subspaces of the display space, and places them at the optimum positions according to the evaluation results.

HeiankyoView defines the order of traversing subspaces as follows. First it specifies the subspace that encloses the center of the display space. This paper describes the subspace as [*s*,*t*], if it is *s*-th along x-axis, and *t*-th along y-axis. Here, x-coordinates of four corners of the subspace [*s*,*t*] are x_s and x_{s+1}, and y-coordinates of them are y_t and y_{t+1}. When the center of the display space is enclosed by the subspace [*s*,*t*], HeiankyoView first traverses it, and then traverses others in the spiral order, as shown in Figure 4(left). For example, it traverses [*s-1*,*t-1*] to [*s-1*,*t+1*], and then traverses [*s-1*,*t+1*] to [*s+1*,*t+1*], and repeats similar traverses.

This spiral order makes faster to find positions where satisfies both conditions 1 and 2, because it avoids to extend the display area by placing rectangles interior the display space. Therefore we think that the spiral order accelerates the algorithm.

While traversing subspaces in the spiral order, HeiankyoView tries to place rectangles at four candidate positions in each subspace. It determines if each candidate position satisfies [condition 1] by overlap check with already placed rectangles. Also, it determines if each candidate position satisfies [condition 2] by calculating the extension of display area by the placement of the rectangles.

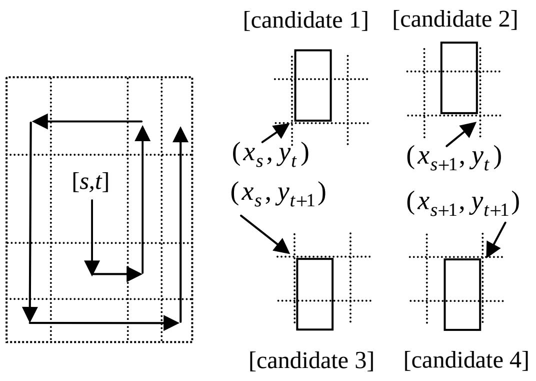

Figure 4. (left) Spiral order of traversing subspaces. (right) Four candidate positions in a subspace.

HeiankyoView calculate positions of four candidates according to the following procedure. Let us denote the width and height of currently placing rectangle as (*w*,*h*), x-coordinates of vertices of the subspace [*s*,*t*] as x_s and x_{s+1}, and y-coordinates of vertices of the subspace as y_t and y_{t+1}. HeiankyoView then tries to place the current rectangle so that one of the vertices of the rectangle locates at one of the following four candidate positions. Here, let us describe two x-coordinates and two y-coordinates of four vertices of a rectangle as $[x_a, x_b, y_c, y_d]$. As shown in Figure 4(right), coordinates of four vertices of the candidates are described as follows:

candidate 1: $[x_s, (x_s + w), y_t, (y_t + h)]$
candidate 2: $[(x_{s+1} - w), x_{s+1}, y_t, (y_t + h)]$
candidate 3: $[x_s, (x_s + w), (y_{t+1} - h), y_{t+1}]$
candidate 4: $[(x_{s+1} - w), x_{s+1}, (y_{t+1} - h), y_{t+1}]$

4.4 Evaluation of candidate positions

Let us denote one of the candidate placement of the rectangle shown in Section 4.3 as $[x_a, x_b, y_c, y_d]$. Here HeiankyoView specifies the integer values i, j, k, l that satisfy the following equations representing the relationship between the four values x_a, x_b, y_c, y_d and positions of vertices of subspaces:

$$x_i \le x_a \le x_{i+1},$$
$$x_j \le x_b \le x_{j+1},$$
$$y_k \le y_c \le y_{k+1},$$
$$y_l \le y_d \le y_{l+1}.$$

Figure 5(left) shows the relationship between above values. HeiankyoView then determines if at least one of the subspaces [*i*,*k*] to [*j*,*l*] have been filled by previously placed rectangles. If there is at least one filled subspace, HeiankyoView determines that the candidate placement cannot satisfy [condition 1], and tries to place the rectangle at other candidate positions. Otherwise, it calculates the extension of display area due to the placement of the rectangle. If the placement does not extend the display area, HeiankyoView decides to place the rectangle there. If the extension of the display area is not zero but smaller then any of previously tried candidate positions, HeiankyoView registers the candidate position as the temporal position. If there is no position that satisfies [condition 1] and does not extend the display area, HeiankyoView decides to place the rectangle at the temporal position.

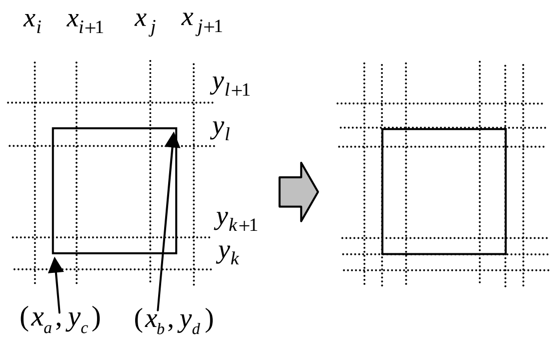

Figure 5. (left) Subspaces that HeiankyoView determines the overlap between rectangles. (right) Subdivision of subspaces.

Deciding the position of the current rectangle, HeiankyoView subdivides the subspaces by the extension lines of the rectangle, as shown in Figure 5(right).

5. Example

Figure 6 is an example of hierarchical data visualization by HeiankyoView. The represented data denotes an archive of file system of an author, which contains 1067 files and 67 directories. Here, files are converted into leaf-nodes, and directories are converted into non-leaf-nodes, to form the hierarchical data. Computation time was about 0.1 second by IBM ThinkPad A31p (Intel Pentium III, 1.7GHz) with Windows 2000 and Java Development Kit (JDK) 1.3.1.

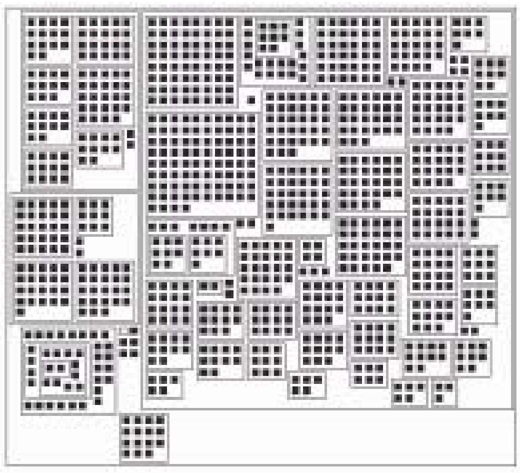

Figure 6. Example.

6. Conclusion

This paper presented a new hierarchical data visualization technique HeiankyoView, which represents the data as nested rectangles, using an orthogonal display layout algorithm.

The study presented in this paper is in a very beginning stage and we just developed a prototype and tried to represent some data. The following issues are therefore our next works on this study:

- Comparison with existing techniques, especially Data Jewelry Box [Ito03] and Quantum Treemap [Bed02], by using some numerical metrics including aspect ratios of rectangles, total display areas, and computation times.
- Extension for stable representation of time-varying data, like as Data Jewelry Box extended for time-varying data [Yam03]. It will be one of the main advantages of HeiankyoView against existing hierarchical data visualization techniques.

- Experiments with applications, like as Data Jewelry Box has been applied to Web sites, Web query results, data mining results, network intrusion detection, and distributed process monitoring.
- Speed up of the algorithm. The number of subspaces of display area is $O(n^2)$ in the worst case, where n is the number of rectangles. The computation time of HeiankyoView is therefore $O(n^2)$ in the worst case.

References

[Bed02] Bederson B., Schneiderman B., Ordered and Quantum Treemaps: Making Effective Use of 2D Space to Display Hierarchies, *ACM Transactions on Graphics*, Vol. 21, No. 4, pp. 833-854, 2002.

[Car95] Carriere J., et al., Research Report: Interacting with Huge Hierarchies Beyond Cone Trees, *IEEE Information Visualization '95*, pp. 74-81, 1995.

[Ito03] Itoh T., Yamaguchi Y., Ikehata Y., and Kajinaga Y., Hierarchical Data Visualization Using a Fast Rectangle-Packing Algorithm, *IEEE Transactions on Visualization and Computer Graphics*, in process.

[Joh91] Johnson B., et al., Tree-Maps: A Space Filling Approach to the Visualization of Hierarchical Information Space, *IEEE Visualization '91*, pp. 275-282, 1991.

[Koi95] Koike H., Fractal Views: A Fractal-Based Method for Controlling Information Display, *ACM Trans. on Information Systems*, 13, 3, pp. 305-323, 1995.

[Lam96] Lamping J., Rao R., The Hyperbolic Browser: A Focus+context Technique for Visualizing Large Hierarchies, *J. Visual Languages and Computing*, 7, 1, 33-55, 1996.

[Rek93] Rekimoto J., The Information Cube: Using Transparency in 3D Information Visualization, Third Annual Workshop on Information Technologies & Systems, pp. 125-132, 1993.

[Spr00] Sprenger T. C., et al, H-BLOB: A Hierarchical Visual Clustering Method Using Implicit Surfaces, *IEEE Visualization 2000*, pp. 61-68, 2000.

[Yam03] Yamaguchi Y., Itoh T., Visualization of Distributed Processes Using "Data Jewelry Box" Algorithm, *CG International 2003*, pp.

```
0) Select rectangles one-by-one.

1) Select subspaces of the display space in the spiral order.

2) Calculate four candidate positions for each subspace, and repeat (2-1) to (2-4) for each candidate. After processing all four candidates, return to 1) and process other subspaces.

(2-1) Try to place the rectangle at the candidate, and
  determine the overlap with previously placed
  rectangles.
(2-2) If the current rectangle overlaps with at least one of other rectangles, return to 2) and process other candidates.
(2-3) If the current rectangle does not overlap with any of other rectangles, and the placement does not extend the display area, decide to place the rectangle there, and go to 3).
(2-4) If the current rectangle does not overlap with any of other rectangles, and the extension of display area is the smallest, register the candidate as temporal position.

3) Place the rectangle, and subdivide the subspaces by extension lines of the edges of the rectangle. Return to 0).
```

Figure 7. Pseudo-code of HeiankyoView.