**Treecube+3D-ViSOM: Combinational Visualization Tool for Browsing 3D Multimedia Data**

Seiji Okajima and Yoshihiro Okada  
Graduate School of Information Science and Electrical Engineering, Kyushu University  
744 Motooka, Nishi-ku, Fukuoka, 819-0395, JAPAN  
{seiji.okajima, okada}@i.kyushu-u.ac.jp

**Abstract**

This paper proposes a new visualization tool for browsing 3D multimedia data. This is realized as a combinational visualization tool of Treecube and 3D-ViSOM which are both proposed by the same research group of the authors. Treecube is a visualization tool for hierarchical information developed as a 3D extension from a 2D visualization tool, Treemap proposed by Ben Shneiderman, et. al in 1992. Treecube is useful for browsing 3D multimedia data stored in a file system because the file system has a hierarchical structure. However, if many data exist in one directory, it is not easy for the user to find his/her required data from it. On the other hand, 3D-ViSOM is a 3D-SOM (Self Organizing Map) based visualization tool for browsing 3D multimedia data. Using the 3D-SOM layout, similar feature data are located in the same area and it is easy for the user to find his/her required data by the browsing. Since 3D-ViSOM can solve the problem Treecube has, the authors propose a combinational visualization tool of Treecube and 3D-ViSOM in this paper.

**Keywords:** Visualization, Browser, Multimedia, 3D-SOM, Treecube, 3D-ViSOM

**1. Introduction**

In this paper, we propose a new visualization tool for browsing 3D multimedia data, which is realized as a combinational visualization tool of Treecube [1,2] and 3D-ViSOM [3]. Recent advances in hardware technologies have made it possible to create 3D images in real time even using a standard PC and 3D CG has become very common in game and movie industries. As a result, many polygonal models and motion data have been created and stored. 3D CG creators and designers need any tools that help them to easily find their required polygonal models and motion data. This fact motivated us to propose a new visualization tool called Treecube for browsing 3D multimedia data originally stored in a file system.

Our Treecube is derived from a 2D visualization tool called Treemap [4] proposed by Ben Shneiderman, et al in 1992. Treemap visualizes hierarchical information. Generally, hierarchical information is represented as a tree structure. Treemap hierarchically lays out each node as a bounding box, whose size is the same as the specific weight or attribute value given to the node. Practically, a lot of tree-structured data exist and the size of such data is going to be greater and greater. Such a huge size of tree-structured data needs an efficient visualization tool. As a result, Treemap has become one of the very useful visualization tools. However, layout algorithms Treemap has are all 2D. Therefore, we have developed a 3D visualization tool, Treecube by extending the 2D layout algorithms of Treemap. This tool is useful for browsing 3D multimedia data, i.e., 2D images, 3D polygonal models, motion data etc., originally stored in a file system because it can automatically lay out them in a specified, restricted 3D space. However, if there are many 3D multimedia data in one directory, it is difficult for the user to find his/her required data efficiently even using Treecube. So, in this paper, we propose a combinational visualization tool of Treecube and 3D-ViSOM because 3D-ViSOM can solve the above problem of Treecube.

Our 3D-ViSOM is a 3D-SOM (Self Organizing Map) based visualization tool for browsing 3D multimedia data. Using the 3D-SOM layout, similar feature data are located in the same area and it becomes easy for the user to find his/her required data by the browsing. Fortunately, both visualization tools Treecube and 3D-ViSOM are realized using IntelligentBox [5], which is a component-based 3D graphics software development system and it is easy to combine them. Hence we developed a combinational visualization tool of Treecube and 3D-ViSOM. In this paper, we show its usefulness as a browser for 3D multimedia data.

The remainder of this paper is organized as follows: Section 2 introduces our Treecube visualization tool. Especially we explain its layout algorithms and interfaces. Section 3 introduces our 3D-ViSOM visualization tool. Especially, we explain feature vectors for each type of 3D multimedia data input for 3D-SOM. Section 4 shows our
proposed combinational visualization tool of Treecube and 3D-ViSOM for browsing 3D multimedia data. Finally Section 5 concludes the paper.

2. **Treecube Visualization Tool**

For Treemap, besides the original layout algorithm called slice-and-dice, there are several extensions, i.e., squarified Treemap [6], ordered Treemap [7] and strip Treemap [8]. Moreover, quantum Treemap [8] is a quantization version of these extensions. We have extended 2D versions of layout algorithms of slice-and-dice, ordered, strip and quantum Treemap to their 3D counterparts. In this section, we explain the outline of those Treecube layout algorithms. For the details of Treecube layout algorithms, see the paper [1].

2.1 **Slice-and-dice Treecube**

The slice-and-dice Treecube algorithm creates a layout where the bounding box of each item (node/leaf) is located in the simple linear order. The algorithm divides a rectangular solid along x, y and z direction and recursively. Figure 1 (left) shows a layout example generated by the slice-and-dice Treecube algorithm. The system visualizes the file and subdirectory structure of a certain directory of a file system. Most bounding boxes shown in the figure are so thin that it is difficult to put 3D multimedia objects in them. This is the same negative feature as that of the slice-and-dice Treemap algorithm. The aspect ratio of each bounding box should be close to one for the visual efficiency.

2.2 **Ordered Treecube**

The ordered Treecube algorithm divides a rectangular solid based on a concept of “pivot”. The pivot node is selected from a node list by a certain criteria. And the remaining nodes are laid out in order to make the pivot into a cubic shape. Figure 1 (center) shows another screen shot of the layout generated using the ordered Treecube algorithm. Although there are still some thin bounding boxes but most bounding boxes have a near cubic shape in comparison with those of Figure 1 (left).

2.3 **Strip Treecube**

The strip Treecube algorithm once divides a given rectangular solid into a set of flat solids called “slice”, and moreover divides each slice into a set of slender solids called “strip”. Each node of a node list is put in the corresponding of strips and its shape is varied into a cubic shape with keeping the order of a hierarchical structure. In Figure 1 (right), the strip Treecube algorithm generates the almost same number of lower aspect ratio-bounding boxes as that of the ordered Treecube algorithm. The strip Treecube algorithm arranges given data simply in a certain direction, so the order of data is almost preserved in comparison with ordered Treecube layout.

2.4 **Quantum Treecube algorithms**

As mentioned above, bounding boxes generated by the Treecube layout algorithms have arbitrary aspect ratios. As extensions of Treemaps, quantum Treemaps [8] were proposed to visualize fixed size objects such as 2D images grouped together into semantic categories. Similarly, we extended our Treecubes to lay out categorized 3D objects, i.e., polygonal models, motion data and 2D images. Our quantum Treecube algorithm generates the layout of multiple fixed size 3D objects. For this, it is necessary to change the size of a rectangular solid in order to make its width, height and depth be integer multiples of the given 3D object size. In other words, a bounding box has a grid of cells in the inside of itself, in each of which each fixed size 3D object is located. Even if there are empty cells in a box, all objects can be laid out sufficiently in the box. However, it may occur undesirably that the volume of a
box quite differs from the total amount of the volumes of objects laid out in the box. For instance, when you want to lay out 50 objects that have \((1 \times 1 \times 1)\) size into a bounding box whose size is \((100 \times 100 \times 100)\), the box resized becomes completely different from the original shape. For avoiding this problem, we use a relative ratio of the width, height and depth of a given rectangular solid instead of their absolute values. Then it is possible to keep out from the drastic deformation of a bounding box by making the number of objects laid out along each x-, y- and z-direction match to the size specified by the user.

2.5 Browsing 3D multimedia data by Treecube

We developed an actual visualization tool for browsing 3D multimedia data, e.g., 2D images, 3D polygonal models, motion data and so on. The Treecube algorithms demonstrated in Figure 1 automatically lay out 3D objects into a specific, restricted 3D area. Figure 2 shows the screen shot of an actual visualization tool that displays 3D multimedia data laid out by the quantum strip Treecube algorithm. Actually this visualization tool is developed using IntelligentBox [5]. As IntelligentBox is a component ware and provides various functionalities as software components called boxes, a file selection functionality for choosing a directory of a file system and an input functionality for entering several parameters necessary for Treecube layout algorithms are implemented as composite boxes. We implemented interactive interfaces and introduced them into the visualization tool for browsing 3D multimedia objects and finding required objects efficiently. Next sub-section introduces such interfaces and explains their functionalities.

2.6 Interactive Interfaces of Treecube

This sub-section treats interfaces implemented to enhance the ability of Treecube as an interactive visualization tool for 3D multimedia data.

![Fig. 3: Example of extraction operation.](image)
hierarchical structure of 3D multimedia data by the operations like the backward/forward of Internet browser.

2.6.2 Cutting plane interface

Treecube is a 3D visualization tool and it has the occlusion problem. In our Treecube, many leaf node objects are put inside of a restricted 3D rectangular solid space and it is hard to see the objects behind other objects. To solve this problem, we introduced a particular interface called “cutting plane”. As shown in Figure 5, the objects before a cutting plane are automatically hidden by the system, and then the objects behind the plane can be seen easily. Its manipulation is very simple. The user only changes the distance of a cutting plane from the eye position because the plane is automatically directed towards the eye point. The user can also change the visibility of a cutting plane by a mouse-click operation on the plane.

As explained in this section, our Treecube visualization tool is useful for browsing 3D multimedia data. However, as previously explained, if there are too many data in one directory of a file system, it is hard for the user to find his/her required data even using Treecube. To solve this problem, another visualization tool called 3D-ViSOM is available and them we combined its functionality into Treecube visualization tool. In the next section, we introduce our 3D-ViSOM visualization tool.

3. 3D-ViSOM Visualization Tool

SOM proposed by T. Kohonen [9] is one of the neural network algorithms. Usually SOM maps high dimensional data records that have more than two attributes onto a 2-dimensional space by analyzing them using their feature vectors. This is called 2D-SOM. In addition to 2D-SOM, there are 1D-SOM and 3D-SOM can perform 1D and 3D mapping respectively. We consider that it is natural to use 3D-SOM for the visualization of 3D multimedia data and we developed 3D-SOM based visualization tool using IntelligentBox. In the following sub-sections, we explain which features of 3D multimedia data, i.e., polygonal models, motions and 2D images, are used for the 3D-SOM layout.

3.1 Feature vector for polygonal model data

We chose shape distribution [10] as the feature vector of polygonal models for the 3D-SOM layout. This information is originally proposed to be used as similarity
measure for the polygonal model search. Shape distribution data is obtained by the sampling of distances between two random points on the surface of a polygonal model. This data is represented as one histogram and we use this histogram as the feature vector of the polygonal model. Figure 6 (a) shows a 3D visualization example of our polygonal model database including 280 models categorized into 14 classes, i.e., Robot, Human, Chair, Sofa, Table, Plant, Door, Car, Glass, Fruit, Pot, Tire, Fish and Head. Similar shaped models are located closely to each other so it is easy to find required polygonal models associatively.

3.2 Feature vector for motion data

We chose feature information obtained by the space division quantization method [11] used as similarity measure for the motion data search. This information means how long each joint exists in each divided region of a 3D space in a complete motion. See the paper [11] for its detail. This information is represented as one set of histograms and we use this set as the feature vector of the motion. Figure 6 (b) shows a 3D visualization example of our motion database including 61 motions categorized into 7 classes, i.e., Kick, Sit, Walk, Throw, Jump, Tumble and Arise. In this case, there is another particular component of IntelligentBox called MotionBox that has functionalities to read a motion file and to display it as its skeleton animation. In the same way as the case of polygonal model data visualization, similar motions are located closely to each other so it is possible to find required motions visually and interactively.

3.3 Feature vector for 2D image data

As for 2D images, we use Hue component of HSV color as the feature information. The feature vector can be obtained as the probability distribution of Hue intensities over a whole image. Figure 6 (c) shows a 3D layout example of our 2D image database including 164 images categorized into 10 classes, i.e., Cat, Dog, Flower, Airplane, Mountain, Sky, Water, Space, Squirrel and Animal.

3.4 Hierarchical browsing of 3D multimedia data

If there are a huge number of data, it is impossible to display and browse them. To deal with this problem, 3D-ViSOM system employs a hierarchical display mechanism.

Fig. 6: 3D-SOM layouts of polygonal models (a), motion data (b) and 2D image data (c) respectively.

Fig. 7: (a) Treecube visualization of many 3D multimedia data, and (b) 3D-ViSOM layout of motion data.
If there are too many data in a small region to see and to access a target object, the system displays only one average, center position data as the representative of those data. When the user selects the average data, the system displays all of the data included in that region by the 3D-SOM layout.

4. **Treecube+3D-ViSOM Visualization Tool**

Finally, this section shows Treecube+3D-ViSOM visualization tool. Figure 7 (a) show an actual Treecube visualization tool for displaying many 3D multimedia data stored in a file system. If there are too many data in one directory, the user can not find his/her required data. For example, the left lower part of Treecube in Figure 7 (a) includes many motion data existing in one directory. In the case like this, using 3D-ViSOM, the user can obtain 3D-SOM layout of those motion data as shown in Figure 7 (b). Similar motion data come to be located in the same area so that the user can find his/her required motion data more easily. As explained in previous sections, Treecube and 3D-ViSOM are both one particular component of IntelligentBox and it is possible to combine them together easily. By combining them, our Treecube comes to have another advantage of 3D-SOM based visualization tool and it became more and more useful for browsing 3D multimedia data.

5. **Concluding Remarks**

In this paper, we proposed a new visualization tool for browsing 3D multimedia data. This tool is realized by combining our already proposed two visualization tools, Treecube and 3D-ViSOM. Originally these visualization tools are developed as individual components of IntelligentBox and it is possible to combine them easily so we developed a combined tool of them. The Treecube algorithms automatically lay out 3D objects, which are originally stored in a file system, in a specified, restricted 3D space. This paper explained these layout algorithms of Treecube and showed that this tool is useful for browsing 3D multimedia data. However, if there are too many data in one directory, even using Treecube, it is difficult for the user to find his/her required data rapidly. This problem of Treecube was possible to be compensated by using 3D-SOM layout mechanism of 3D-ViSOM. So, we introduced 3D-ViSOM visualization mechanism into Treecube visualization tool. As a result, our Treecube combined with 3D-ViSOM became more powerful visualization tool for browsing 3D multimedia data.

As future works, we have to introduce more practical interfaces into Treecube+3D-ViSOM visualization tool for more efficient browsing of 3D multimedia data. We also have to check the usefulness of our visualization tool by consulting actual users who use it.

**References**


