Journal of Zhejiang University SCIENCE A
ISSN 1009-3095 (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Out-of-core clustering of volumetric datasets

GRANBERG Carl J., LI Ling
(*Department of Computing, Curtin University of Technology, Perth, Australia*)
E-mail: granberg@cs.curtin.edu.au; ling@cs.curtin.edu.au
Received Apr. 7, 2006; revision accepted Apr. 19, 2006

Abstract: In this paper we present a novel method for dividing and clustering large volumetric scalar out-of-core datasets. This work is based on the Ordered Cluster Binary Tree (OCBT) structure created using a top-down or divisive clustering method. The OCBT structure allows fast and efficient sub volume queries to be made in combination with level of detail (LOD) queries of the tree. The initial partitioning of the large out-of-core dataset is done by using non-axis aligned planes calculated using Principal Component Analysis (PCA). A hybrid OCBT structure is also proposed where an in-core cluster binary tree is combined with a large out-of-core file.

Key words: Out-of-core clustering, Hybrid rendering, Scientific visualization
doi:10.1631/jzus.2006.A1134 **Document code:** A **CLC number:** TP39

INTRODUCTION

The size of the average dataset used in the technical and medical community today is ever growing. It seems that the graphics capabilities of mainstream computers are always a few steps behind the requirements for visualizing such large datasets. This dilemma creates the need for multiresolution rendering systems that allow the datasets to be rendered at multiple levels of detail (LOD). For datasets whose sizes range up to a couple of gigabytes, the main memory of mainstream computers is not enough to contain them. It then becomes necessary to store and render these datasets from secondary memory i.e. the hard drive. The methods dealing with such datasets are referred to as out-of-core methods. Much research has been done in this area, most often concerning very large triangular meshes. Our research however focuses on out-of-core volumetric scalar datasets and the construction of a multiresolution structure from them. Our primary focus is on medical datasets. The recently developed Ordered Cluster Binary Tree (OCBT) structure can be used to query the out-of-core dataset for a sub volume at a selected LOD efficiently.

RELATED WORK

Most research on the rendering of volumetric datasets is focused on various texture-based methods. Many of these techniques utilize octrees to divide volumes into smaller entities. However we chose to build our multiresolution system on another technique since octrees restrict the volume splitting to axis aligned planes. Our research is based on the hierarchical data structure and cluster algorithm instead. There are two approaches to hierarchical clustering: top-down and bottom-up. The top-down approach of hierarchical clustering starts with a cluster containing all data points that are split recursively until the whole cluster tree has been created. Heckel *et al.*(1999b) used this approach to cluster vector fields. The cluster with the highest error value was split into two sub clusters with the aim to reduce the error values of the resulting sub clusters. The bottom-up approach starts with all points as individual points and merges the two clusters with the least difference until one big cluster has been formed. Telea and van Wijk (1999) used this approach to simplify complex vector fields using an elliptic similarity function. They merged the pair of vectors with the least position, magnitude and

direction difference until all vectors have been merged into one single vector. In 1997, Heckel *et al*.(1997; 1999a) reconstructed meshes from scattered points using an adaptive clustering method that locates nearly planar points in the dataset. These subsets created tiles and the gaps in-between the tiles were filled using constrained Delaunay triangulation. In 2003, Co *et al*.(2003a) presented a technique for clustering of volumetric scalar fields. They utilized a clustering method building on the work of Heckel *et al*.(1997; 1999a). Their algorithm split clusters using Principal Component Analysis (PCA) in a non-axis aligned fashion. The cluster's error values were created using a multiquadric Radial Basis Function (RBF). Initially all data points were added to a single cluster that was split iteratively in a top-down fashion to generate a Cluster Binary Tree (CBT). Recent work by Granberg and Li (2005) has improved upon the CBT structure presented by Co *et al*. to allow efficient spatial queries and faster clustering. This new data structure is called the OCBT. At each OCBT node the splitting plane calculated with PCA is stored and used for spatial queries. The tree structure is similar to that of a Binary Space Partitioning (BSP) Tree or a k-D Tree. For more in-depth details on the construction of these cluster binary trees please refer to (Co *et al*., 2003a; Granberg and Li, 2005).

ORDERED CLUSTER BINARY TREE OVERVIEW

The OCBT is created with a top-down (divisive) clustering method. A cluster is defined as a set of points P, described as $P=\{x, y, z, F\}$ where x, y and z are the spatial dimensions and F is the scalar value associated with that position. Initially a single cluster is created containing all the points from a volumetric scalar dataset. This cluster is iteratively divided using non-axis aligned splitting planes calculated using PCA. Each splitting plane cuts through the volume and divides it into two sub clusters. This repeated splitting generates a hierarchical cluster structure where the cluster that was split is the parent of the two resulting sub clusters. Each cluster is assigned an error value which is calculated using for example a radial basis function. The clusters available for splitting are stored in a priority queue, which is ordered according the cluster error values. The cluster at the top of the queue (i.e. the cluster with the highest error value) is always selected for splitting. The cluster splitting increases the homogeneity of the resulting sub clusters, thus the average error value in the priority queue is decreased. The splitting can stop either when the queue is empty or the highest cluster error value is under a set threshold. For more information on the creation of cluster binary trees please refer to (Co *et al*., 2003a; Granberg and Li, 2005).

SPLITTING LARGE DATASETS IN MANAGEABLE PARTS

As mentioned in the previous section, the OCBT cluster algorithm uses PCA to split clusters in a non-axis aligned fashion. The PCA calculates a splitting plane that divides a cluster through the cluster center so that the error value of the resulting children is decreased. However a problem appears when this operation needs to be performed on an out-of-core dataset. In order to allow this clustering algorithm to work out-of-core, the original dataset must be partitioned into manageable parts. Each part can then be read in and processed in-core. At first glance it might seem a good idea just to partition the dataset into small regular cubes that fits in the internal memory (similar to Octree splitting).

However such scheme produces undesirable results later during the construction of the cluster binary tree. Partitioning the dataset initially this way creates areas around the edges of the small cubes with much higher cluster error values than normal during the clustering stage. It results in a much higher number of points being distributed to the edges of the cubes that results in poor clustering, as shown in Fig.1 (see page 1139) where both images show the virtual human foot dataset with approximately 8000 particles resulting from a level-of-detail query. Note that Fig.1b has a much more evenly distribution of points than Fig.1a as is the expected result from an LOD-query. It can be seen that using regular splitting is not suitable for such application to partition a large dataset into smaller parts, since it results in poor quality clustering. Instead, the splitting planes created by PCA are used when partitioning out-of-core datasets. However, by performing the PCA directly on an

out-of-core dataset, the dataset must be accessed an unfeasible number of times.

The splitting planes are calculated from the eigenvectors of the clusters. These in turn are calculated from the cluster covariance matrices. This fact can be used to devise a scheme to partition large datasets too large to fit in internal memory. In Figs.2a~2d, all four images represent the same dataset at different resolutions. Consider the covariance matrices of these images (for more information on the calculation of these covariance matrices please refer to (Co *et al*., 2003a; Granberg and Li, 2005)):

$$\boldsymbol{CM}(a)=\begin{bmatrix}0.3317 & 0.2483 & 0.1744\\ 0.2483 & 0.3316 & 0.1339\\ 0.1744 & 0.1339 & 0.1539\end{bmatrix},$$

$$\boldsymbol{CM}(b)=\begin{bmatrix}0.3300 & 0.2467 & 0.1741\\ 0.2469 & 0.3300 & 0.1335\\ 0.1741 & 0.1335 & 0.1508\end{bmatrix},$$

$$\boldsymbol{CM}(c)=\begin{bmatrix}0.3235 & 0.2402 & 0.1720\\ 0.2402 & 0.3235 & 0.1314\\ 0.1720 & 0.1314 & 0.1509\end{bmatrix},$$

$$\boldsymbol{CM}(d)=\begin{bmatrix}0.2879 & 0.2045 & 0.1599\\ 0.2045 & 0.2879 & 0.1189\\ 0.1599 & 0.1189 & 0.1387\end{bmatrix}.$$

Note how covariance matrices ***CM***(a), ***CM***(*b*) and ***CM***(*c*) are almost identical, and only ***CM***(*d*) is a bit different. Fig.2d contains 100 data points compared to the original image with its 90000 data points. Even though Fig.2d contains only 0.11% of the number of data points compared to the original image, it still produces a fairly accurate covariance matrix. The splitting planes used in the cluster algorithms (Co *et al*., 2003a; Granberg and Li, 2005), are obtained completely based on the covariance matrix of a dataset. Therefore, calculating the splitting plane from a scaled down version of the dataset will generate a valid splitting plane applicable to the original dataset as well. The large out-of-core dataset is hence linearly scaled down to fit into the main memory, so that the splitting plane calculations can be performed on the in-core version of the dataset. These splitting planes can then be applied to the original out-of-core dataset.

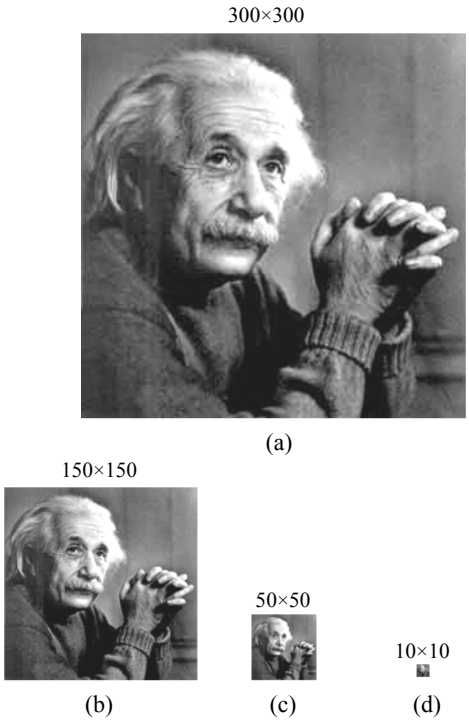
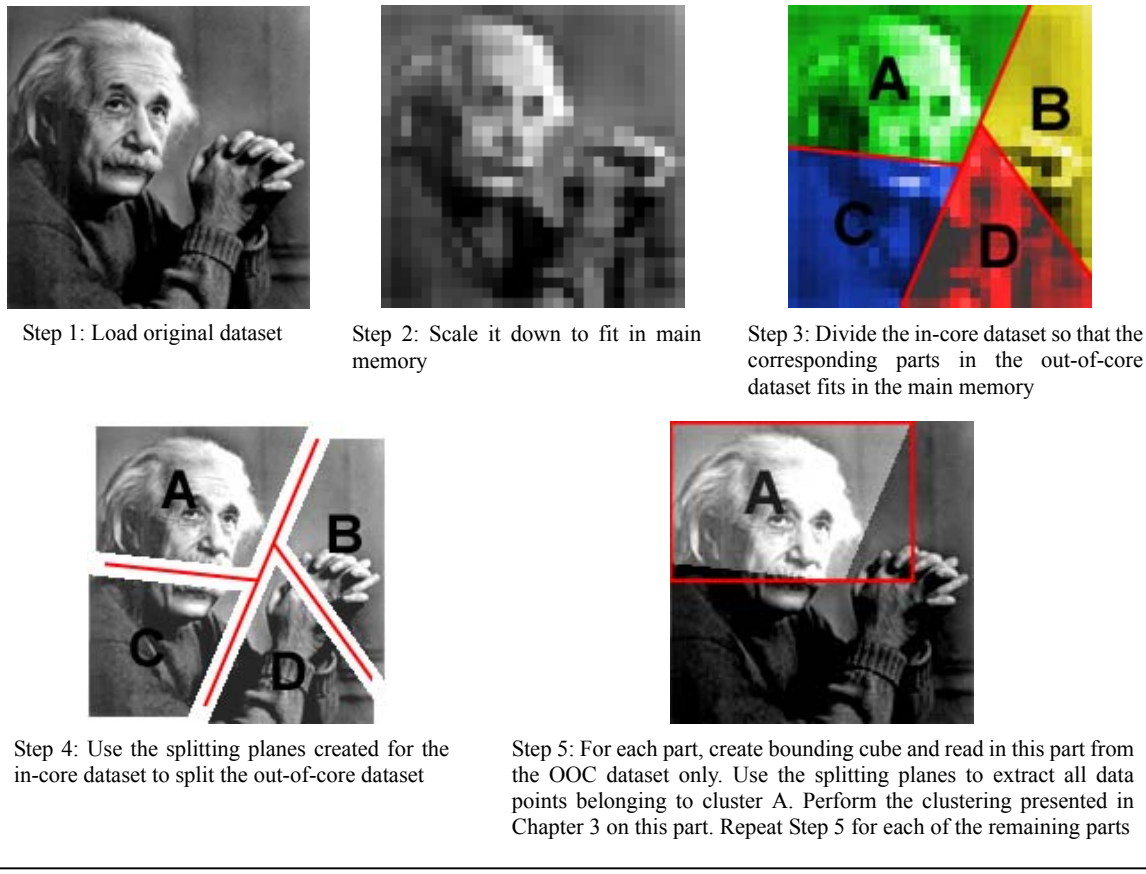


Fig.2 (a) The original image, (b)~(d) The scaled down versions of Image (a)

CREATING AN ORDERED CLUSTER BINARY TREE OUT-OF-CORE

Large datasets must be divided into manageable parts that can be read into the main memory and processed there as presented in previous work (Co *et al*., 2003a; Granberg and Li, 2005). First the original dataset is read in and scaled down to fit into the main memory. The scale factor is stored, which represents how many voxels in the original dataset are represented by one voxel in the scaled down dataset. This scale factor is used to determine when enough splits have been performed so that all the divided parts fit into the main memory. Fig.3 provides an overview of the out-of-core clustering procedure.

Fig.3 shows the process of dividing a large out-of-core dataset using non-axis aligned splitting planes generated from a scaled down version of the original dataset, instead of just using simple octree splitting. Due to the overhead of this data structure the resulting OCBT needs to be stored out-of-core as well. The structure of the out-of-core OCBT is shown in Fig.4 (see page 1139). Note that the fields A~D in Fig.4 represents the segmented fields shown in Fig.3 respectively.



Step 1: Load original dataset

Step 2: Scale it down to fit in main memory

Step 3: Divide the in-core dataset so that the corresponding parts in the out-of-core dataset fits in the main memory

Step 4: Use the splitting planes created for the in-core dataset to split the out-of-core dataset

Step 5: For each part, create bounding cube and read in this part from the OOC dataset only. Use the splitting planes to extract all data points belonging to cluster A. Perform the clustering presented in Chapter 3 on this part. Repeat Step 5 for each of the remaining parts

Fig.3 The procedure for dividing an out-of-core dataset into manageable parts

The out-of-core OCBT hence contains two parts as showed in Fig.4. One part is stored in-core and the other is stored out-of-core. The in-core part is saved as a normal OCBT with the slight difference that the leaf nodes are pointers to locations in a long continuous file stored out-of-core as shown in Fig.5 (see page 1139). The tree creation process can be very slow especially if the dataset is huge. The in-core part of the tree is stored in a separate file so that after the tree has been processed it can be reloaded in a matter of seconds.

The in-core part of the structure should contain as many nodes as possible, since traversing the data structure in-core is 100~1000 times faster than traversing it out-of-core. During the initial partitioning, only a minimum number of splits are performed until each partition of the dataset to fit into the main memory. For large datasets this number of splits may range from 10~50 splits. However, ideally around a million OCBT nodes should be stored in the finished in-core part of the OCBT. This is done by cropping the OCBT for each partition at a certain depth. The cropping depth determines how many nodes is stored in-core, while the rest of the nodes are stored out-of-core. Once a small partition has been clustered into a sub-tree (for example the partitions A~D in Fig.4) it is cropped at a predetermined depth. All the nodes with depth greater than the cropping depth are stored in the out-of-core file, while the rest are stored in the in-core file. When the in-core part of the tree is traversed and a leaf node is reached, the search continues in the out-of-core file at the correct location. For each leaf node in the in-core tree a file position is saved indicating where the nodes of that particular sub tree start and the length in bytes, as shown in Fig.5. A sequential read of all the nodes in a sub tree is often faster to perform on the secondary memory than accessing only the nodes of interest at random. By cutting down the information stored in each of the out-of-core nodes the overhead for the data structure

is greatly reduced. Splitting planes or the bounding sphere information are no longer stored in the out-of-core part of the tree. This saves 5 float values (20 bytes) for each node, which has a huge impact when several million points are stored. It is not efficient to traverse the out-of-core OCBT file in the same fashion as the regular OCBT since searching the regular OCBT relies on random memory access.

OUT-OF-CORE QUERIES

The out-of-core OCBT is queried in much the same way as the regular OCBT with a small number of changes made in the implementation to accommodate the new data structure. As stated in the previous section the splitting planes or the bounding sphere hierarchy are no longer stored in the out-of-core part of the OCBT. This means that any form of queries performed on the out-of-core file must check whether each point traversed matches the search criteria. Without the information stored with the splitting planes and bounding spheres, there is no longer a good way to reduce the amount of nodes that must be searched, nor is there an early search termination. Instead a brute force method is relied upon. By cutting down the overhead of each out-of-core node the number of nodes that can be processed is increased significantly and as stated before, a sequential read is much more efficient than a random access read anyway. This also means that many more nodes may be stored using the same disc space as before.

The in-core part of the tree is queried exactly as presented in (Granberg and Li, 2005) using the splitting plane information stored in each node to divide the search space into two parts. This is done every time the search algorithm visits an OCBT node. The in-core part of the tree is the intelligent part where the search space is decimated to a very small part. When searching for a specific point, the in-core part of the OCBT divides the search space into as many parts as there are leaf nodes in the tree. At each leaf node there is an array (stored in the out-of-core file) containing a known number of points, among which one is the point of interest. The whole array in the out-of-core file is read sequentially and each point is compared against the search criteria. The more information is stored in the in-core part of the tree, the smaller the out-of-core arrays will be and the less information will have to be read from the out-of-core file to find the queried data points. As discussed before, random access is used to move the read position randomly to the correct start position in the out-of-core file (which is very slow). Sequential reads are then used to read in the entire array (i.e. all data points in a certain sub tree), which is much faster to perform on the secondary memory.

RESULTS

The experiments were performed on a Pentium IV 3.0 GHz PC with 1 GB memory and a nVidia GeForce FX 5200. The out-of-core OCBT is visualized as suggested in (Granberg and Li, 2005). This hybrid rendering method consists of a rough texture-based rendering for an overview of the dataset and a high-quality particle-based rendering method for the selected region of interest.

Head dataset

The Head dataset is a subset of the Virtual Human Female color dataset and consists of 618×674×603 voxels, all with 3 bytes of color data. From this we create a scaled down version with 128^3 (approximately 2 million) voxels that we use for the texture-based rendering and to create the initial splitting planes. It took 7 h and 27 min to create the whole out-of-core OCBT. The original dataset is 735 MB in size and the resulting OCBT is 1.82 GB in size. Fig.6 (see page 1139) shows the visual results of the Head dataset. The area of interest consists of particles created from a spatial query performed during run-time from the out-of-core OCBT. The query took 4 s to perform and returned 525000 voxels. This dataset can be visualized in interactive frame rates (5~20 frames per second). The frame rate depends on the number of voxels the query returns. The number of nodes in the created OCBT is approximately 116000000 nodes.

Torso dataset

Fig.7 shows the visual renderings of the Torso dataset (1608×785×300 voxels). From this Figure, it can clearly be seen how the 128^3 texture-based rendering provides a rough representation of the entire dataset while the particle-based rendering increases

the detail of the selected area. The number of nodes in the created OCBT is approximately 164000000. The clustering of this dataset took 11 h 19 min.

(a) (b)

Fig.1 The virtual human foot dataset clustered using two different ways of splitting the dataset into manageable parts. Image (a) is generated with initial regular splitting. The resulting unwanted effect is circled in red. Image (b) is generated with splitting planes calculated by PCA

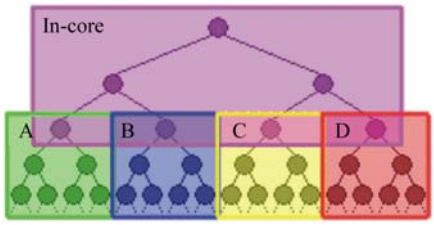


Fig.4 The structure of the out-of-core Ordered Cluster Binary Tree

CONCLUSION AND FUTURE WORK

We have presented a method to create an out-of-core multiresolution data structure that can be queried and used to visualize large volumetric scalar datasets. With this method we can create a cluster tree from far larger datasets than presented by previous work. Co *et al*.(2003a) presented a cluster tree with 600000 nodes, which was clustered in 44 min using 4 Pentium IV PCs (three 2.8 GHz, 2 GB memory, and one 2.2 GHz, 1 GB memory). Granberg and Li (2005) improved on the clustering speed and were able to cluster 1.8 million nodes in 27 min using a Pentium IV 3.0 GHz PC with 1 GB memory. By implementing the methods presented in this paper we have created cluster binary trees as large as 164000000 nodes. There is however still much room for improvement of this clustering algorithm. One improvement would be to create a more visually pleasing rendering system of this data structure. This could for instance be a ray-tracing or a splatting system. Further optimizations of the clustering process and the out-of-core file structure should also be explored.

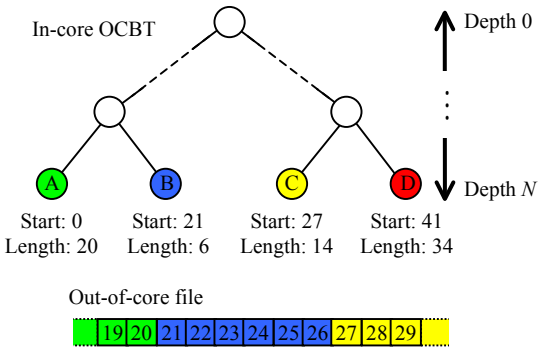


Fig.5 The connection between the leaf nodes in the in-core OCBT and the out-of-core file

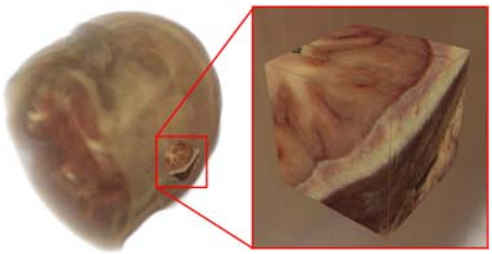


Fig.6 The Virtual Human Female Head dataset visualized using the out-of-core OCBT. The total size of the OCBT is 1.82 GB

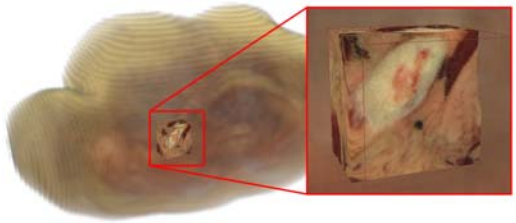


Fig.7 The Virtual Human Female Torso dataset visualized using the out-of-core OCBT. The area of interest consists of 325000 voxels, queried in 11 s. The total size of the OCBT is 2.43 GB

References

Co, C.S., Hagen, H., Hamann, B., Heckel, B., Joy, K.I., 2003a. Hierarchical Clustering for Unstructured Volumetric Scalar Fields. Proceedings IEEE Visualization'2003,

p.325-332.

Co, C.S., Hamann, B., Joy, K.I., 2003b. Iso-Splatting: A Point-based Alternative to Isosurface Visualization. Proceedings of the 11th Pacific Conference on Computer Graphics and Applications 2003, p.325-334. [doi:10.1109/PCCGA.2003.1238274]

Ding, C., He, X., 2002. Cluster Merging and Splitting in Hierarchical Clustering Algorithms. Proc. of the 2002 IEEE International Conference on Data Mining (ICDM'02), p.139-146. [doi:10.1109/ICDM.2002.1183896]

Granberg, C., Li, L., 2005. Hierarchical Clustering of Large Volumetric Datasets. Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia, p.425-428. [doi:10.1145/1101389.1101473]

Heckel, B., Uva, A., Hamann, B., 1997. Cluster-Based Generation of Hierarchical Surface Models. Sientific Visualization'97, p.105-114.

Heckel, B., Uva, A., Hamann, B., Joy, K.I., 1999a. Surface Reconstruction Using Adaptive Clustering Methods. Technical Report CSE-99-1, Computer Science Department, University of California, Davis.

Heckel, B., Weber, G., Hamann, B., Joy, K., 1999b. Construction of Vector Field Hierarchies. Proceedings IEEE Visualization'99, p.19-25.

Isenburg, M., Gumhold, S., 2003. Out-of-core compression for gigantic polygon meshes. *ACM Trans. Graphics (SIGGRAPH'03)*, **22**(3):935-942. [doi:10.1145/882262.882366]

Ma, K., McCormick, P., Wilson, B., 2002. A Hardware-Assisted Hybrid Rendering Technique for Interactive Volume Visualization. Volume Visualization and Graphics Symposium 2002, p.123-130.

Masciari, E., Pizzuti, C., Raimondo, G., Talia, D., 2001. Using an Out-of-core Technique for Clustering Large Datasets. Proceedings DEXA 2001 Workshops. IEEE Computer Society Press, p.133-137.

Telea, A., van Wijk, J., 1999. Simplified Representation of Vector Fields. Proceedings IEEE Visualization'99, p.35-42.

Pavan, M., Pelillo, M., 2003. Dominant Sets and Hierarchical Clustering. Proc. of the 9th IEEE International Conference on Computer Vision, p.362-369. [doi:10.1109/ICCV.2003.1238367]

Yoon, S.E., Salmon, B., Gayle, R., Manocha, D., 2005. Quick-VDR: Out-of-core view-dependent rendering of gigantic models. *IEEE Transactions on Visualization and Computer Graphics*, **11**(4):369-382. [doi:10.1109/TVCG.2005.64]