Guihua Shan · Maojin Xie · Feng’An Li · Yang Gao · Xuebin Chi

Interactive visual exploration of halos in large-scale cosmology simulation

Received: 30 April 2014 / Revised: 5 July 2014 / Accepted: 6 July 2014 / Published online: 29 July 2014
© The Visualization Society of Japan 2014

Abstract Halo is one of the most important basic elements in cosmology simulation, which merges from small clumps to ever larger objects. The processes of halos’ birth and merging play a fundamental role in studying the evolution of large-scale cosmological structure. In this paper, a visual analysis system is developed to interactively identify and explore the evolution histories of thousands of halos. In this system, an intelligent structure-aware selection method in What You See Is What You Get manner is designed to efficiently define user’s interesting region in 3D space with 2D hand-drawn lasso input. Then the exact information of halos within this 3D region is identified by data mining in the merger tree files. To avoid visual clutter, all the halos are projected in 2D space with MDS method. Through the linked view of 3D view and 2D graph, users can interactively explore these halos, including the tracing path and the evolution history tree.

Keyword Point cloud · Halo exploration · Merger tree · Visual analysis

1 Introduction

In cosmology, dark matter halo is a cluster of dark matter whose density exceeds the threshold. During large-scale cosmology simulation process, even-distributed dark matter tends to group together under gravity and forms small halos. Small halos collide and merge with each other and form ever larger halos. Halo is one of the most fundamental elements in large-scale simulation. In N-body simulation, halos can be found with Friend-Of-Friend (FOF) method (Riebe et al. 2011). A merger tree is a tree-like structure which records a halo’s merging history. It is a special tree. A tree node may contain several halos and the most massive halo in the node is named as the master halo, while the others are called satellite halos. A merger tree records the local evolution process of the universe, which lays an important position for merger tree.

G. Shan · M. Xie · F. Li · Y. Gao · X. Chi
SuperComputing Center, Computer Network Information Center, Chinese Academy of Sciences, Beijing, China
E-mail: sgh@sccas.cn
M. Xie
E-mail: xiemaojin@sccas.cn
Feng’An Li
E-mail: Lifa@sccas.cn
Y. Gao
E-mail: Gaoyang@sccas.cn
X. Chi
E-mail: chi@sccas.cn

J Vis (2014) 17:145–156
DOI 10.1007/s12650-014-0206-5
Large-scale cosmology simulation generates large numbers of halos, which range from thousands to millions. Large numbers of halos and their merger trees make things more sophisticated and bring great challenges in halo exploring and merging history back tracing.

To show all of the tracing paths or merger trees together at one time for all particles or halos is impossible, so it is important to interactively explore the interesting objects and study the physical processes; selection of interesting particles from visualization context is essential for interaction. The selected interesting area can be further explored, which not only helps the user focus on the interesting object, but also dramatically reduce the processing data volume to make further exploration more efficient. For example, an astronomer prefers to select an interesting halo structure from current view to trace back its evolution or the related statistic information.

Although there is lots of visualization work for cosmological data, most of them focus on rendering of particles data (Hassan and Fluke 2011; Lipsa et al. 2012), where the interactive functions are limited to real-time rendering and no further information feeds back except the continuous changes of the scene.

Manually selection on the rendering image with 2D mouse allows users to obtain a spatial area by multiple operations. But it is hard for people to define an exact 3D surface to encircle the desired spatial region with 2D input, which is tedious and time consuming. Moreover, when an interesting halo is found, they may want to know not only its properties like mass, radius and velocity, but also its evolution history or the merging location both in space and time. A more intelligent and efficient interaction with large-scale temporal dark matter data is needed.

In this paper, we propose an interactive visual analysis system. The workflow of the system is illustrated in Fig. 2. A WYSIWYG particle selection method is provided, in which Cloud-in-Cell (CIC) method is adopted to generate grid data and then marching cubes are employed to gain isosurface and use projected area are used as criteria to automatically filter the user’s most wanted particle cluster. With our method, a 3D region is easy to define with a hand-drawn 2D lasso, which makes the selection process of halos in 3D space quite efficient. And then we propose our visual analysis scheme. Several 2D/3D views are organized as linked view style and halo information is shown in different views. 2D layout for halos makes halo size clear. Multidimensional scaling (MDS) is employed in 2D layout to avoid overlapping and its distance-reserved feature makes the halos’ relationship more clear. We use 2D and 3D layout for merger trees. Dark matter origin and sub-halos moving path are more clear in 3D layout while merge time is easier to perceive in 2D layout. The main contributions of the paper are:

1. An intelligent structure-aware selection method in WYSIWYG manner is designed, which provides an intuitive and efficient way to interact with large-scale data (Fig. 1)
2. 2D halo layout with MDS projection avoids the visual clutter, which allows an efficient choice for the interesting halo
3. With the interaction in three spaces (3D view, 2D projected graph and merge tree), the system provides a visual analysis for deep exploration of halos. In the remainder of the paper, firstly we review the
2 Related work

Visualization and visual analysis, which extract and present useful information out of petabytes of cosmology data, are essential for astronomy discovery. There are many such works in the past decade, both in dark matter visualization, evolution history visualization and object selection.

2.1 Dark matter visualization

Particle based simulation data usually employ some kinds of splatting approaches and approximate the spatial distributions of physical quantities using kernel interpolation techniques. Interactive large-scale dark matter visualization was proposed by Fraedrich et al. (2009) who employed hierarchy quantization and data compression to reduce data volume. GPU acceleration is used for decompress on the fly. Interactive visualization of time-varying data set was proposed by Hopf and Ertl (2003), which adopts out-of-core for data accessing and B-splines for position interpolation.

Except splatting approaches, volume-rendering based dark matter visualization was also proposed by Fraedrich et al. (2010). Particle sets are represented in a level-of-detail manner and perspective grid is employed to allow effectively reducing the amount of primitives to be processed in run time. Kaehler et al. (2012) proposed a novel approach for cosmic web visualization which makes use of full phase-space information to generate a tetrahedral tessellation of the computational domain, with mesh vertices defined by the simulation’s dark matter particle position.

As Hassan and Fluke (2011) and Lipsa et al. (2012) mentioned, respectively, many visualization works for cosmology simulation focused on single time step, while temporal data are usually visualized in batch mode. Although some interactive visualization work has been proposed as mentioned previously in this session, most interactive work is restricted to real-time rendering, without any more information feeding back except the continuous changes of the scene. This is of vital importance in interactive exploration, but is still far from being satisfactory for scientists to explore more detail information and make discovery when they are working on the scene, for example, when an interesting halo is found, they may want to know not only its properties like mass, radius and velocity, but also its evolution history, etc.

2.2 Evolution history exploration

Di Matteo et al. (2008) visualized black holes with arrows indicating their masses and location in multiple constant images. Takle et al. (2012) performed tracking and building halo evolution history with merger tree in parallel and show the merge tree in 2D. Fluke et al. (2009) developed S2PLOT, a novel multipurpose visualizing tool for interactively visualizing cosmology data, including 3D merger tree. By adopting VRML, they were able to interchange the interactive 3D scientific visualization among a variety of mediums. The structure is quite clear as the redshift is high, but overlaps severely as redshift decrease. There is lack of an intuitive global view to select a target halo at redshift $z = 0$, which is the most important snapshot for the domain scientists as it is the present cosmic world.

In exploratory visualization, linked view (Roberts 2004) is efficient as there are more than one variables and analysis aspects at one time. Each of the variables or aspects shown in one view makes things clearer, and the linked view makes the interaction respond in all views.

2.3 Selection of interesting objects

For more intuitively and efficiently selecting the interesting area in 3D data, many approaches are developed such as ray casting-based methods (Peng et al. 2010; Kohlmann et al. 2012), surface-based methods (Argelaguet and Andujar 2009). Guo et al. (2011) first presented a novel “WYSIWYG” idea for volume rendering. With intuitive user interface like Photoshop instead of traditional transfer function edition, users can easily achieve satisfied volume rendering results. Later, more work in WYSIWYG style was provided, Guo and Yuan (2013) described a local WYSIWYG volume rendering by constructing contour tree in the preprocess, where more useful gestures are also designed for easily controlling the volume rendering.
Wiebel et al. (2012) presented a novel raycasting based method, which allows user to intuitively select spatial position in volumetric renderings. Observable structures are characterized by large jumps in the accumulated opacity; the picked structure corresponds to the largest jump of the accumulated opacity. Yu et al. (2012) described a lasso-like technique, where the grid is constructed in the box of selected area and marching cubes are used for enclosing the particle cloud structure intuitively lasso on the interesting 2d project.

We also propose a WYSIWYG technology for the particle cloud selection in our system. Compared to (Yu et al. 2012), our method is able to separate the different enclosed structures of selected spatial particles, and the structure which is perceived most is chosen by default. (Yu et al. 2012), the selected enclosed structures are unable to be separated, which is inconvenient for further analysis. Compared to (Wiebel et al. 2012), the data type and the principle to realize the “what you see is what you get/pick” is different. In our research, we work on particle cloud data instead of volume data. We use marching cubes and flood fill to get the interesting structure, instead of volume rendering and opacity.

3 The WYSIWYG selection algorithm

The WYSIWYG particle cloud selection algorithm can automatically select the particle cluster user interested according to the drawn region. The input of the algorithm is all the particles and their attribute data. Our algorithm consists of the following five steps:

1. Preliminarily marking the particles that are projected in the user-drawn region.
2. Gridding the marked particles and getting a regular volume data.
3. Extracting particles in clusters with larger density through the marching cubes algorithm.
4. Splitting the clusters into independent ones by an improved flood fill algorithm.
5. Obtaining the particle cluster interested by computing and comparing each cluster’s projective area. By default, we suppose that the point cluster with the largest projective area is the most interesting one. Figure 2 shows the processing flowchart of the algorithm.

3.1 Marking the particles in the user-drawn region

We provide two kinds of drawn region: one is circle and the other is polygon. According to the drawn lasso, we generate a filled polygon and render it to the frame buffer. We name the rendered image a mask. Using the mask, we project all the particles in this view and mark the particles in the drawn region. The following algorithm steps are all based on the marked particles. Figure 3 shows two kinds of user-drawn lasso.

![Fig. 2 The processing flow chart of the WYSIWYG algorithm](image-url)
3.2 Constructing grid for marked particles

We use the isosurface extraction algorithm marching cubes to select the particle clusters which have dense distribution. Before executing the marching cubes algorithm, we construct a regular grid of the marked particles and compute the density of each node. As a scalar field of the volume data. According to the range of the marked particles’ coordinate, we obtain an axis aligned bounding box, as shown in Fig. 4a. We construct a volume data and split it into \( n_x \times n_y \times n_z \) cells as Fig. 4b shows.

To apply marching cubes to extract isosurface, we calculate particle density \( D(x, y, z) \) at each node \((x, y, z)\) as the scalar field. We calculate it as:

\[
D(x, y, z) = \sum_{i=0}^{m} (d_{ix} + d_{iy} + d_{iz})
\]

Where \((x, y, z)\) is a grid node; \(m\) is the total number of particles inside the eight cells which take \((x, y, z)\) as their vertex; \(d_{ix}, d_{iy}, d_{iz}\) represent the contribution of each particle \(i\) inside the eight cells to the node \((x, y, z)\) in \(x, y, z\) axis, respectively. That is, the density of each grid node is the sum of the contribution from all particles in the eight cells that the node belongs to. Actually, we calculate each particle’s contribution to the eight vertices of which cell that the particle belongs to. By accumulating the contribution at each vertex, we get the density of each node.

If the coordinate of the particle \(p\) is \((p_x, p_y, p_z)\), the boundary of the volume data in \(x, y, z\) axis is \((\text{min}_x, \text{max}_x), (\text{min}_y, \text{max}_y), (\text{min}_z, \text{max}_z)\), the length of each cell is \(l_{\text{cell}}\), then the relative coordinate of \(p\) for each direction \(k = x, y, z\) can be calculated as

\[
r_k = (p_k - \text{min}_k)/l_{\text{cell}}
\]
Rounding up and down to \( r_k \), we obtain two nodes coordinate \( k_1 \) and \( k_2 \) closest to particle \( p \) in direction \( k \):

\[
\begin{align*}
k_1 &= 0, k_2 = 1; (r_k = 0) \\
k_1 &= \text{floor}(r_k), k_2 = \text{ceil}(r_k); (0 < r_k < N_k + 1) \\
k_1 &= N_k; k_2 = N_k + 1; (r_k = N_k + 1)
\end{align*}
\]

where \( N_k \) is the cells number in axis \( k \).

According to the distance between the particle and the node, we can calculate the contribution of the particle to its adjacent grid nodes in axis \( k \):

\[
w_{k1} = k_2 - r_k, w_{k2} = r_k - k_1
\]

The contribution of particle \( p \) to the eight grid nodes of the cell where particle \( p \) is inside can be expressed as follows:

\[
\text{contri}(x_o, y_p, z_q) + = w_{x_o} + w_{y_p} + w_{z_q}, o = \{1, 2\}, p = \{1, 2\}, q = \{1, 2\}
\]

3.3 Surface extraction based on marching cubes

The application of surface extraction algorithm can divide the particles into clusters according to the distribution intensity. First, we calculate the average density \( \rho_0 \) of the grid nodes as the initial threshold, which could be adjusted in real time during visualization. Applying the marching cubes algorithm to the grid, we seek out regions with density \( \rho \geq \rho_0 \). The result of marching cubes algorithm is surfaces whose densities equal to \( \rho_0 \), and the purpose of this article is to select particles. Therefore, we need to mark the particles inside the isosurfaces. The marching cubes algorithm is processed in voxel. According to the comparison result between the density of each voxel’s eight vertices and the threshold, we classify the voxels as voxels inside the isosurface, voxels outside the isosurface and voxels on the isosurface. All of particles in the inner voxels must be located inside the isosurface; particles in the outer voxels must all be located outside the isosurface; particles in the boundary voxels require part in and part out of the isosurface.

\[
\begin{align*}
\text{Cell value}_i &= 0, \text{outer Voxel} \\
&= [1, 254], \text{boundary Voxel} \\
&= 255, \text{Inner Voxel}
\end{align*}
\]

By tagging the voxels with the tags above, the isosurfaces and their inner particles can be easily mapped to the volume data, and then we can extract the particle clusters. To extract particles in the volume, we need to build a lookup table which records the mapping relation between the voxel and the particles inside. This lookup table can be made at the same time during grid density calculation. For inner voxels, we mark all the particles according to the lookup table. However, the boundary voxels requires further judgment for particles inside. We calculate the density for each particle inside the boundary voxels. Comparing this density with the threshold, we distinguish whether it is inside the isosurface or not. Like calculating density for grid nodes, we should determine the voxel index the particle belongs to according to the particle’s coordinate. The particle’s density can be obtained as cubic interpolation of the voxel’s eight vertices. If the particle density value is greater than the threshold, the particle is accounted to be inside the isosurface and marked; otherwise, it is outside and then abandoned.
3.4 Clusters splitting

Now, we have obtained all the clusters with density greater than the threshold. To separate the clusters, we extend the flood fill algorithm from 2D to 3D to get the connected components through the filling area. Finally, the clusters independent. The flood fill algorithm is commonly used to obtain connected components in 2D images. However, this algorithm requires seed to be assigned. In this paper, we do the filling in voxels. We choose the first voxel in the isosurfaces as the first seed. In the process of filling, we mark the visited voxels to ensure that the filling process won’t revisit voxels.

The conditions to be filled are as follows:

1. The current voxel’s coordinates are within the legal scope, i.e.,
   \[ x \in (0, N_x), y \in (0, N_y), z \in (0, N_z) \]  \hspace{1cm} (7)

2. The current voxel has not been visited.

3. The current voxel is located inside or on the boundary of the isosurfaces.

When the voxel satisfies the above three conditions at the same time, it will be marked with the ClusterId (a variable of identity of the cluster). For the particles inside the voxel, they will be all marked with the same cluster id if the voxel is an inner voxel; or only the particles whose density is greater than the threshold will be marked with cluster id when the voxel is a boundary voxel. Considering the six-neighborhood of the seed voxel in filling processing, if at least one vertex of the voxel has density greater than the threshold, we fill the voxel and mark the particles following the principle above. After finishing one connected space, we will find the next seed voxel for next flood filling, which is marked as ClusterId + 1, and repeat the above operation, until all voxels are marked as visited. Finally, we will obtain the segmentation.

However, when two different clusters are too near, in the case that their boundaries are in two adjacent voxels or even in a same voxel, which is illustrated in Fig. 5, they will be mistakenly marked as one cluster. In Fig. 6, the purple parts are the isosurfaces, and the yellow and green parts mark the particles in the isosurfaces. Figure 6a shows only yellow part which means that there is one cluster in the scene. Obviously, there are two clusters. So, filling conditions is not right for this case.

We improve the filling conditions in flood fill algorithm, when marching toward each direction, we do the following judgment:

The next voxel has at least one side in the marching direction satisfying the condition that both vertices’ grid node density is greater than the threshold. For example, the marching direction is \( x + 1 \), and the next voxel with its vertices is tagged as Fig. 7. The four edges in the marching direction \( x + 1 \) are: e0, e1, e2, e3. If only densities of vertex 3 and vertex 2 are greater than the threshold, then the voxel will be filled.

(a) Wrong split result  \hspace{1cm} (b) Right split

Fig. 6  Split results in different filling conditions
If the condition above is true, then fill the next voxel and selectively mark the particles inside.

When comparing the grid node density of each voxel with the threshold in the marching cubes algorithm, we record the comparison result between both vertices’ density of voxel’s four edges with threshold in \(x/y/z\) directions as Tag. So Tag is an integer range from 0 to 7. Each bit of Tag stores two marching directions’ result. At least one edge satisfies the condition that both vertices’ density is greater than the threshold, and then this bit is signed as 1. If all the 4 edges in the marching direction can not satisfy the condition, this bit is signed as 0. With the filling algorithm, we make a decision to continue or stop depending on the Tag value. With this improved algorithm, we get the correction segmentation in Fig. 6b.

3.5 Hardware accelerated computation of projected areas

We have separated the different particle groups in Sect. 3.4. For scientists, it is preferable to automatically select out the particle cluster that they are most interested in. We assume that the cluster occupying the largest projection area in the user-drawn region is the one scientists most wanted. This, of course, is true in most cases since they really perceive the largest projection area when they make a lasso in the rendering image.

Therefore, the default output of the selected cluster should be the one with the largest projection area. To calculate each cluster’s area, the vertices data of each isosurface is required. So not only the particles inside should be split, but also the isosurface encircling each cluster should also be stored, respectively. At the time we do the marching cubes for each voxel, one lookup table recording the voxel id and vertices information of the isosurfaces in it is built, as shown in Fig. 8a. When splitting the clusters using flood fill algorithm, we store the isosurfaces’ vertices according to the clusters, respectively, and then get another lookup table as Fig. 8b shows.

Using the ClusterId as index of vertex sequence of isosurfaces, we render every encircled isosurface to the frame buffer. Figure 9a shows the isosurface generated by user selection in the current view direction. Figure 9b shows the original isosurfaces and particles in the same view direction. Since we render all isosurfaces in the same view in their original positions, the result projection is the overlapped area. Changing the viewpoint, we get the separated projection as shown in Fig. 10a. Projecting the isosurfaces according to their locations is not desirable, because there are two isosurfaces in this case.

To avoid overlapping of each isosurface’s projection, we render them into \(N\) viewports, where \(N\) is the amount of clusters. We split the window along the \(x\) axis into \(N\) viewports and render each isosurface in one viewport. The resulting projection area is squeezed in the \(x\) direction, but all projection areas are squeezed with the same ratio; thus there is no influence on the following area comparison step.

By counting the non-black pixels of each projection, we get the largest cluster and return its ClusterId to the user. Finally, we mark the returned cluster in yellow, as shown in Fig. 11. The two green clusters are independent and can also be extracted by this method.

4 Halo exploration

With WYSIWYG technology, we are able to extract interesting halos efficiently from a chaotic data and analyze its evolution process in linked view style. We will describe our technology implementation and case studies in this chapter.
We performed the experiments on a machine with an Intel® Core™ i5-3470 3.20 GHz CPU, 8 GB RAM and an NVidia Quadro 600 graphics card with 1 GB video memory. The data are from the cosmology dark matter simulation with \( n \)-body method using 1 million particles. The output is a scatter point dataset with 64 time steps, where each particle has 6 variables such as ID, position, velocity, mass, dispersion, and density. The number of halos in all time steps adds up to 536048.
4.1 Halo layout and interaction

Technically, we build a lookup table which records the mapping relationship between the voxels and the halos inside with method similar to Sect. 3.3. The difference is that the center positions of master halos substitute for particle positions.

Halos displayed in 3D space may suffer from overlapping and occlusion, which makes interactive analysis not so convenient. To get a better layout for master halos in 2D view, we employ MDS to map halos’ positions from 3D space to 2D space, while halos’ distances are reserved as much as possible. Thus the halos belonging to the same FOF group will cluster together. We use a disc to represent the halo, with its radius mapping to the halo’s radius, and its color and brightness mapping to velocity dispersion and density, respectively. Thus, we can perceive the main properties of each halo in an intuitive way.

When interacting with a halo, we click on that disc representing it in 2D view, then the halo id is determined by comparing halo center position and the cursor position under Manhattan distance.

4.2 Case study 1: tracing path

In this case study we focus on how to show the particles evolution of the universe in an intuitive way. It is one of the most important problems that astronomers concern. Scientists used animation to study the evolution. However, since human memory is temporary, it is easy to lose information details when the observation is mutative. Another problem that bothered scientists when analyzing the evolution of the universe is that visualizing the time sequence of all data may bring heavy IO burden. Low efficiency has always been the main obstacle for further research in cosmogony. All the problems mentioned above can be solved by using our system. Scientists can select out the most interesting particles by the 2D input lasso. Then we draw an image of the evolution trace path of the selected particles. We use the “cyan–blue–purple–yellow” color table to represent time sequences from far to near on the time line (see Fig. 12 for example). The image alone contains the evolution process of the selected particles. Compared with the animation, our method is more intuitive and needs less IO. In addition to the space trace path, we can clearly see the dark halo merger, divergence and other physical processes in the development of the structure.

4.3 Case study 2: evolution merger tree of a halo

Although trace path is a nice way to express the evolution process in 3D space, the merger tree is a better quantization expression when exploring the merging history, both in mass and time.

We draw the merger tree in a level-wise way, with the level representing time. Disc is used to express the halo, with its size corresponding to halo size, while density and velocity dispersion are encoded into brightness and color. Thus, we can easily perceive the main attributes of each halo in the merger tree. Edges with gradually varied color are used to express the merger relationship.

When we select a region in 3D view, we can retrieve a set of halos. They scatter in the 2D region, with their positions transformed by MDS, while the relative distances are reserved. We can see from Fig. 13a that after two small clusters of particles are selected (shown in yellow), halos inside are shown as discs in 2D in the top right view, with two main clusters, respectively. We can click on the discs; they will be highlighted with another color and their merger tree will be displayed in bottom view. From the merger tree, we can clearly find two main streams (marked as 1 and 2 with red dash circles, which we call stream 1 and stream 2, respectively) merged into the main stream of selected halo.
We can also easily notice that there is no merging after stream 1 and stream 2, but the density and velocity dispersion change from high to lower, which means that the halo collided and some of the mass split away. In Fig. 13b, the different interested cluster is selected, the tracing path a selected root halo is showed in 3D view instead of one time step space view, which illustrate intuitive of the halo evolution in the 3D spatial and temporal axes.

The performance when dealing with the data which has 1 million particles and 536,048 halos is shown in Table 1. In interactive exploration, only in the first stage all the particles are rendered for a globe view. The small interesting number of halos is loaded for temporal analysis after the user-drawn selection. The total rendering time includes particles rendering, MDS projection of the halos in the isosurface, merger subtree searching for the root halo and rendering of the subtree. It ranges from 0.15 to 0.3 s depending on the total number of halos in the subtree. If the selected root halo is merged by more halos, then it takes more time when searching and rendering. From Table 1, we also can see that the merger tree process is time consuming. It takes about 80 % of the total time. This is because when doing the merger tree pass, we need to retrieve the halo data sequence to find out the parent halo recursively. The WYSIWYG selection method takes 0.03 s on average, but this process is executed only at the beginning and not recomputed if a different root halo is selected in the MDS projection graph; so this selection method consumes little time.
5 Conclusion

We designed a cosmology visualization system for halo analysis. An efficient WYSIWYG selection scheme was proposed and halos could be easily selected for further analysis. We adopted MDS in transforming halos’ positions from 3D space to 2D space to reserve the relative distance and get an intuitive 2D layout. The 3D space, 2D graph and 2D merge tree were combined together to support an interactive exploration of halos evolution. In our case study, we analyzed the halo’s evolution process in two ways, one was trace path, with the moving path clearly shown in 3D view, another was merger tree, with the merger process clearly shown in quantities, both in time and mass.

Acknowledgments The authors are grateful to Prof. Feng Longlong from Purple Mountain Observatory, CAS and Doctor Zhu Weishan from Nanjing University for the data set and instructive suggestions. The work was supported by Chinese National Sciences Foundation No. 91230115.

References

Argelaguet F, Andujar C (2009) Efficient 3d pointing selection in cluttered virtual environments. IEEE Comput Graph Appl 29(6):34–43
Di Matteo T, Colberg J, Springel V et al (2008) Direct cosmological simulations of the growth of black holes and galaxies[J]. Astrophys J 676(1):33
Fluke CJ, Barnes DG, Jones NT (2009) Interchanging interactive 3D graphics for astronomy[J]. Publ Astron Soc Austral 26(1):37–47
Friedrich R, Schneider J, Westermann R (2009) Exploring the millennium run-scalable rendering of large-scale cosmological datasets. IEEE Trans Vis Comput Graph 15(6):1251–1258
Friedrich R, Auer S, Westermann R (2010) Efficient high-quality volume rendering of SPH data[J]. IEEE Trans Vis Comput Graph 16(6):1533–1540
Guo H, Yuan X (2013), Local WYSIWYG volume visualization, IEEE pacific visualization symposium (PacificVis 2013), Sydney, NSW, Australia, Feb. 26-Mar. 1., 2013, pp 65–72
Guo H, Mao N, Yuan X (2011) WYSIWYG (what you see is what you get) volume visualization[J]. IEEE Trans Vis Comput Graph (Vis ‘11) 17(12):2106–2114
Hassan A, Fluke C (2011) Scientific visualization in astronomy: towards the petascale astronomy era. Publ Astron Soc Austral 28(2):150–170
Hopf M, Ertl T (2003). Hierarchical splatting of scattered data. IEEE Vis Comput Graph 57
Kaehler R, Hahn O, Abel T (2012) A novel approach to visualizing dark matter simulations[J]. IEEE Trans Vis Comput Graph 18(12):2078–2087
Kohlmann P, Bruckner S, Kanitsar A, Groller ME (2009) Contextual picking of volumetric structures. In: Peter Eades H-WS, Thomas Ertl (eds) Proceedings of the IEEE pacific visualization symposium, pp 185–192
Lipsa, Laramee R, Cox SJ, Roberts JC, Walker R, Borkin MI, Pfister H (2012) Visualization for the physical sciences. Computer graphics forum 31(6):2317–2347
Peng H, Ruan Z, Long F, Myers JH, Myers EW (2010) V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets. Nat Biotech 28(4):348–353
Riebe K, Partl A M, Enke H et al (2011) The MultiDark database: release of the Bolshoi and MultiDark cosmological simulations[J]. arXiv preprint arXiv:1109.0003
Roberts JC (2004). Exploratory visualization with multiple linked views. In: MacEachren A, Kraak MJ, Dykes J (eds) Exploring geovisualization
Takle J, Silver D, Heitmann K (2012), A case study: Tracking and visualizing the evolution of dark matter halos and groups of satellite halos in cosmology simulations. In: IEEE conference on visual analytics science and technology (VAST), Poster, pp 243–244
Wiebel A, Vos FM, Foerster D, Hege H-C (2012). WYSIWYP: what you see is what you pick. IEEE Trans Vis Comput Graph 18(12)
Yu L, Efstatthiou K, Isenberg P et al (2012) Efficient structure-aware selection techniques for 3D Point cloud visualizations with 2DOF input[J]. IEEE Trans Vis Comput Graph 18(12):2245–2254