Using Graphics Hardware for Multiple Datasets Visualization

Gaurav Khanduja
Louisiana State University
Department of Computer Science
USA (70803), Baton Rouge, Louisiana
khanduja@bit.csc.lsu.edu

Dr. Bijaya B. Karki
Louisiana State University
Department of Computer Science
USA (70803), Baton Rouge, Louisiana
karki@csc.lsu.edu

ABSTRACT

We have applied three graphics hardware-based approaches to support concurrent visualization of multiple sets of volumetric scalar data. They include volume rendering, clipping and isosurface extraction methods, which exploit 3D textures and advanced per pixel operations. These methods are expected to give better interactive frame rates for multiple datasets visualization (MDV) compared to the software-based methods. The rendering time in each case increases nonlinearly with the increasing the number (*N*) of the datasets being visualized. We can identify three regimes, which can be characterized by different *time-N* slope value. The first regime with small slope value continues up to about 5 datasets, then the second regime with medium slope value continues up to about 25 datasets, and finally the third regime with much larger slope value continues up to 35 datasets. With volume shading enabled, the rendering time increases on average whereas the transition and maximum *N* values decrease. We propose the dynamic-resolution approach for increasing the maximum *N* and frame rates for above MDV techniques.

Keywords
Multiple datasets visualization, 3D textures, GPU programming, Volume rendering, Clipping, Isosurface extraction, volume spreadsheets

1. INTRODUCTION

Visualization of three-dimensional scalar data has played a crucial role in understanding the behavior of an associated system. However, often for complete insight in the physical system there is a need to analyze different datasets simultaneously of the given system, representing different time step, pressure, temperature. Multiple Datasets Visualization (MDV) [Sch04, Kha05, Kha06a, Kha06b] plays a crucial role in understanding such physical systems. MDV means, rendering two or more dataset in the same visualization either side by side or in fused form. In this paper, we focus on the MDV techniques in light of recent advances in GPU technology and texture based rendering using volume spreadsheets (side-by-side comparison of datasets). Khanduja and Karki [Kha06a, Kha06b] showed the application of MDV for three dimensional scalar volume data. Similar concept for 2D images has been studied widely [Lev94, Chi97, Jan00] for over the decade. Figure 1 shows the visualization [Kar06] for magnesium silicate (post pervoskite) data after clipping.

Primary goal is to achieve interactive visualization for volume scalar data. We present three texture-based approaches exploiting the features of the modern graphics hardware for MDV. The first one is volume rendering involving texture mapping slice by slice with appropriate alpha blending enabled [Cab94]. The second approach is clipping, to uncover important details of a dataset. The planar and box clipping [Kha06b] represents the simple form of clipping based on surface rendering. We also extend the idea of voxelized clipping [Wei03] to MDV. This technique requires an extra 3D texture representing the clip geometry. Third approach is the isosurface extraction using 3D textures. The method proposed here does not require the polygonal representation of the isosurface geometry and thus is applicable to MDV for improving interactive frame rates.. To further improve the effectiveness of above mentioned techniques, volume shading based on the

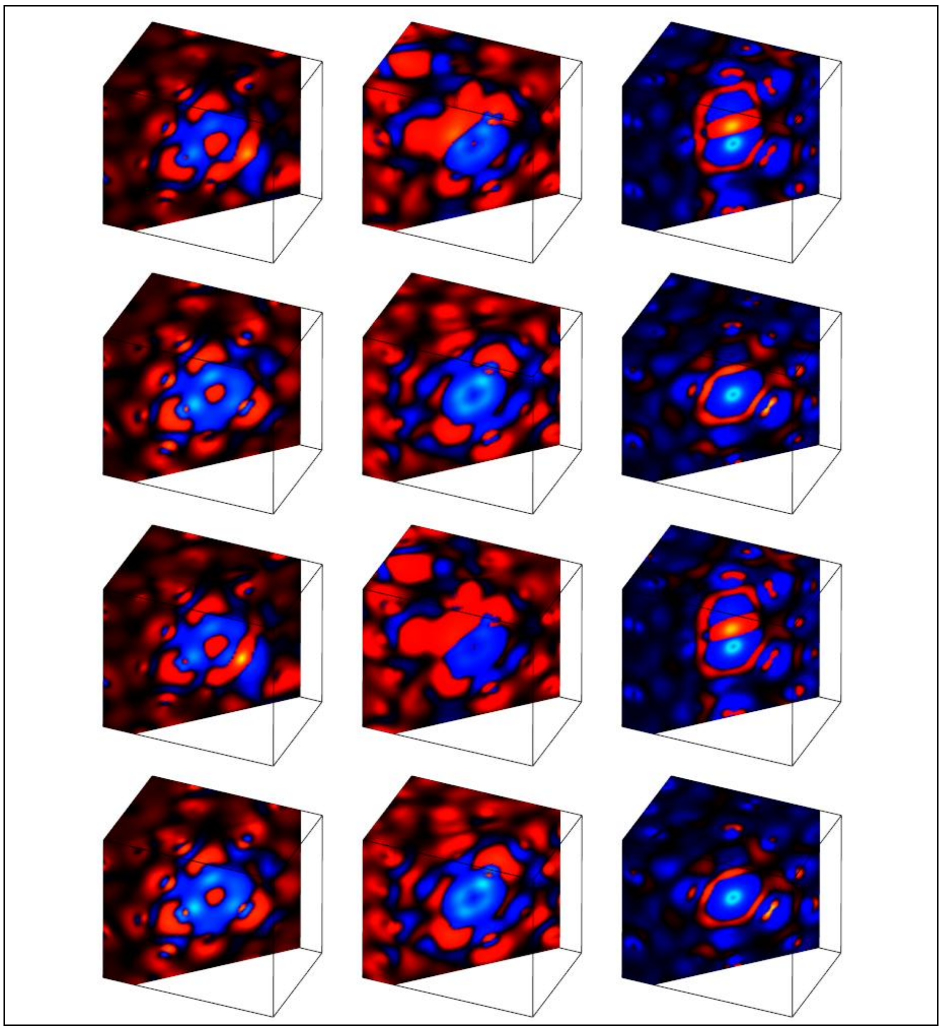


Figure 1: Visualization of electron density difference induced by the Mg (left), Si (centre) and O (right) vacancies in the 60-site $MgSiO_3$ system. First and second rows show final configuration (after atomic structural optimization) for migrating ion and fixed vacancy respectively. Third and fourth rows show the corresponding initial configuration.

phong model [Pho75] is applied in conjunction with the above texture based techniques.

The outline of the paper is as follows. In Section 2, related work is discussed. In Section 3, different texture-based MDV techniques including volume rendering, box clipping, voxelized clipping and isosurface extraction are presented. In Section 4, volume shading is discussed. In Section 5, we discuss dynamic resolution approach. Finally, the Section 6 contains important conclusions and future directions.

2. RELATED WORK

Visualization of three-dimensional scalar data has been area of research for over last two decades. Several visualization methods are available for volumetric scalar datasets [Mei00]. Multiple datasets have been visualized in many occasions [Cru96; Sch04, Kha05, Kha06a, Kha06b]. Schluze [Sch04] uses concept of multiple variable of dataset; all time steps are presented as animation and multiple datasets side by side for MDV. Khanduja and Karki [Kha06a, Kha06b] proposed isosurface extraction using Marching Cube [Lor87] algorithm and texture based approach for MDV. MDV graph (rendering time vs. number of datasets) in [Kha06a, Kha06b], shows non-linear behavior. Both the polygon generation time and the polygon rendering time [Kha06a] show the rapid increase once the swapping between main memory and virtual memory starts occurring due to increase in the number of datasets. Similarly, in [Kha06b], results exhibit non-linear behavior for MDV using textures. [Kha06a, Kha06b] deals with the isosurface and external 3D surface mapping techniques for MDV. The approaches used [Kha06a, Kha06b] in these paper do not utilize capabilities of modern GPU. Moreover texture based clipping [Kha06b] is based on clipping planes and do not discuss complex clipping geometries in context to

MDV. In this paper, we extend these ideas for MDV using hardware accelerated texture based techniques for the MDV. The texture-based techniques include volume rendering, clipping and isosurface extraction methods. Our results show similar non-linear trend. For performance analysis, we consider the 3D textures of charge density distributions in real material systems (MgO and $MgSiO_3$ pervoskite), which are investigated on routine basis by parallel quantum mechanical simulations [Cod05]. The datasets considered are of moderate size e.g. 256^3, which is quite common for current scientific and engineering applications. This paper also presents the effect of shading, which has not been investigated previously in context of MDV.

Texture mapping [Cab94, Cul93, Wil94] is one of the widely used techniques for visualization of scalar data. Often clipping is used as an important tool with texture-based volume rendering for uncovering the hidden details. Clipping planes [Gel96] were previously used for clipping. Weiskopf et al. have proposed techniques for volume clipping with complex geometries, which are based on the depth structure and voxelization of the clip geometry [Wei03]. Some studies have used volume clipping based on stencil test [Wes98], isosurface clipping [For92] and interactive clipping combined with dual-resolution texture-based volume rendering [Kha05].

An isosurface represents a 3D surface at a constant scalar value within a volume. Lot of work for isosurface extraction [Lor87, Cli88, Sch92, Par98, Sch04, Kip05, Geo06] has been done over last two decades. Westermann [Wes98] proposed 3D texture based technique for isosurface extraction. GPU based raycasting [Had05, Kip05] approach has been effectively used for isosurface extraction. A number of techniques for combining 3D texture volume rendering with lighting and shading have been proposed [Gel96, Hau96, Wes98, Wei03]. In [Gel96] the sum of precomputed ambient and reflected light is stored in texture volume and standard 3D texture rendering is performed. In [Hau96] the voxel gradient and volume density are stored together as 3D texture. Due to recent advances of commodity graphics hardware, texture-based rendering is able to achieve acceptable frame-rates with high image quality [Wil94; Wei00; Cul93, Wil02, Wei03].

3. TEXTURE-BASED MDV TECHNIQUES

Transfer Function

Visualization of the scalar volume data requires transfer function for mapping the scalar value to the RGB color space. This section describes the transfer function (Figure 2) used in this paper except for the isosurface extraction where it is not needed, since we are extracting the isosurface corresponding to single isovalue at a given instant. Here R is the scalar value. By changing R_{CUT1} and R_{CUT2}, we can highlight different region in the dataset.

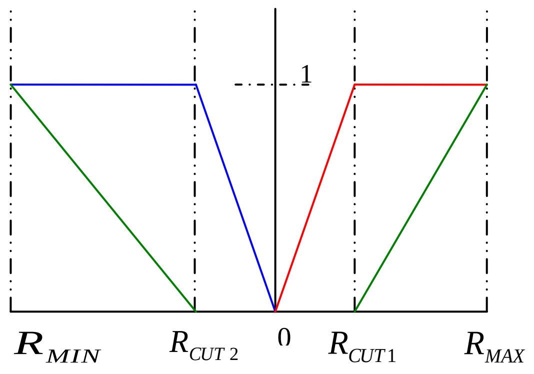


Figure 2: Transfer Function for the Scalar Volume Data. Red, Green and Blue curve represents the transfer function T_R, T_G and T_B respectively. R_{MIN} and R_{MAX} are the maximum and minimum value in the dataset.

We have used linear functions for RGB values, which can be changed to exponential, parabolic depending upon the dataset and the user requirements.

Volume Rendering

The texture mapping approach utilizes 3D textured data slices [Cab94; Cul93; Wil94]. The 3D texture approach can sample the data in the s, t, or r directions freely so the slices are always oriented perpendicular to the viewer's line of sight. Tri-linear interpolation happens at hardware level thus allowing using arbitrary number of slices with an appropriate re-sampling on the slices. Figure 3 shows the volume rendering using 3D textures with alpha blending enabled.

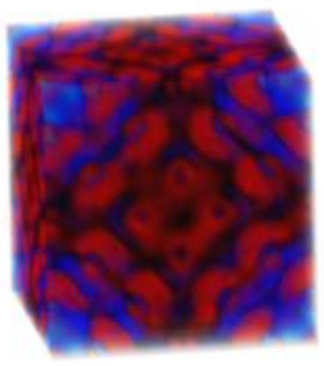


Figure 3: Volume Rendering with Alpha Blending

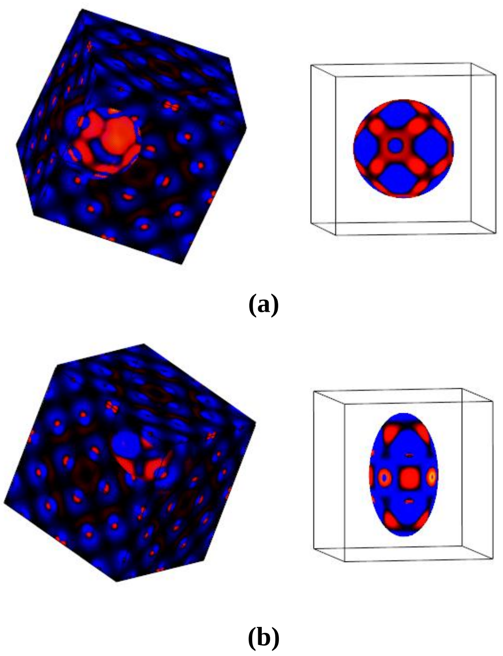
MDV using 3D textures involves following steps. First step involves loading the dataset. Next step is to choose the number of slices perpendicular to the viewing direction for each texture. Third step is to use texture coordinate generation to texture the slice properly with respect to each 3D texture data. In the fourth step, rendering of the texture slices is performed from back to front, towards the viewing position, with appropriate blending performed at each slice. Finally, next dataset is loaded and all of the above steps are repeated for this dataset. For clipping, isosurface extraction and volume shading, the calculations are performed in the GPU before fragments are finally rendered and after the texture and vertex coordinates are bound together in step 4. As the viewpoint and viewing direction is changed, data slice positions need to be recomputed and texture transformation matrix needs to be updated.

Clipping

The purpose of clipping is to explore the hidden details in the dataset. The clipping can be either, volume probing, where fragments lying inside the clipped geometry are kept or volume clipping, where fragments lying outside the clipping geometry are kept. In this section we present clipping based on surface texture rendering (planar and box clipping) [Kha06b] and voxelized clipping approaches which can be interactively used with MDV.

Planar/Box clipping technique utilizes the surface rendering, which involves mapping texture only the visible surface area. Volumetric data is represented by a set of triangles on which the 3D texture is mapped. Initially six planar surfaces of simulation box are rendered. In this approach, we find single or multiples surfaces cutting the volume and then bound the intersecting surfaces in the form of simple polygons. These polygons determine the new set of externally visible surfaces of the volume and the textured data is mapped only on these externally visible polygons. Thus, only the surfaces defined by a set of visible clipping polygons (single or multiple) are rendered. For planar clipping, the intersection of plane with 12 triangles is considered Implementation details of planar and box clipping can be found in [Kha06b].

Voxelized clipping technique [Wei03] utilizes two textures, volume object and clipping object. The 3D texture of the clipping object uses binary representation. The alpha value of the clipping texture is set to 0 or 1 depending upon whether the voxel lies inside the clipping geometry or not. Depending upon the alpha value of the clipping texture GPU assigns the alpha value to the volume texture. The fragments lying outside the clipping texture are assigned value zero. These fragments are discarded using the alpha test for volume probing. For volume clipping the fragments lying outside are assigned the value 1. The clip object can be moved around by mapping the texture coordinates of the clipping texture although changing the shape of the clipping object requires the loading of the new 3D texture. Since the method utilizes the binary representation there is a visible transition between the voxels having value 0 and 1.


(a)

(b)

Figure 4: Voxelized clipping. (a) Spherical Clipping Object. (b) Ellipsoidal Clipping Object

To overcome this artifact induced by the use of the binary textures, we utilize the distance map similar to the one given by Lorenson [Lor93]. Evaluation of $F(x, y, z)$ at any point produces a value < 0, = 0, or > 0, where $F(x, y, z)$ gives the signed Euclidean distance of the voxel from the surface of the clipping object. Points that lie inside the clipping object are assigned negative values and points outside the clipping object are assigned the positive values. The points on the surface of the clipping object have a distance zero. If RGBA 3D textures are used, these distances get clamped to the range (0, 1). Floating-point textures retain the floating-point value stored in them as it is, thus removing any round of error due to clamping in RGBA textures. For different clipping geometries there needs to be a different clipping 3D texture. The same 3D clipping object can be reused for the more than one number of dataset in case of MDV. A new 3D texture needs to be loaded if different clipping geometries are used with different datasets. Figure 4 shows the clipping based on the spherical and ellipsoid clipping object.

Isosurface Extraction

This section presents an isosurface technique, which avoids an intermediate generation of polygonal geometry for the isosurface. Our approach is similar to one proposed by Westermann [Wes98]. Since we can exploit the trilinear interpolation of 3D textures, isosurface (Figure 5) generation using 3D textures is thus equivalent to picking up those voxels through which isosurface passes. Each element in the 3D texture is assigned the scalar value as its alpha component. Then the modified alpha test: (R_{MAX}) > *isoValue* > (R_{MIN}) is applied to the resulting 3D texture to find the voxels containing the isosurface. R_{MAX} and R_{MIN} give the maximum and minimum value respectively for the corresponding voxel, which is used to test whether the isosurface corresponding to the particular isovalue passes through the voxel or not. Modern programmable GPUs can be utilized to perform this task efficiently without the loss of performance. Since the values in the 3D texture are clamped to the region [0, 1]; this approach might not pick up all the areas for the resulting isosurface or the isosurface may not be continuous.

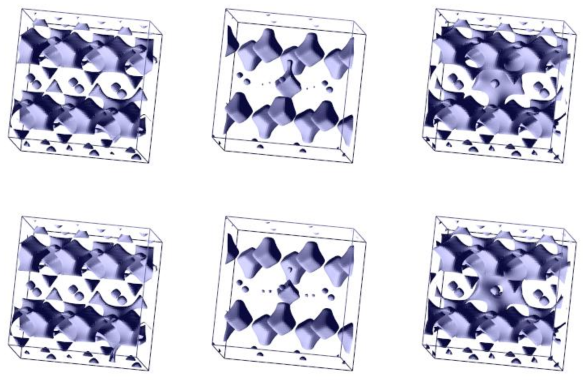


Figure 5: Isosurface Extraction of $MgSiO_3$ showing Mg, Si and O vacancy from left to right with isovalue 0.021. Top layer shows the isosurface after the optimization and lower layer shows the isosurface before the optimization

To overcome this problem we use the floating point 3D textures as described above. For isosurface generation we can use 16 or 32 bit floating point 3D textures. Most of the modern GPU's provide trilinear interpolation for 16 bit floating point 3D textures. This functionality is not available for the 32 bit floating point textures. Therefore, for using 32 bit floating point 3D textures for the isosurface extraction, trilinear interpolation needs to be implemented at the GPU level.

Rendering Time for MDV Techniques

Our analysis involves the rendering time for volume rendering, clipping and isosurface extraction with and without volume shading in context of MDV. Figure 6 shows the rendering time as a function of number (*N*) of datasets, which are visualized concurrently.

In case of volume rendering, alpha blending is enabled. For volume rendering, we get the frame rate of 15.6 fps for single dataset, which gradually fall with increasing number of datasets. Rendering time starts to increase rapidly after 5 dataset. In case of box clipping using surface rendering, we get better frame rates as compared to the volume rendering time. With increasing number of datasets rendering time increases but still remains low as compared to regular volume rendering. Even for 10 datasets the frame rate is above 3, which is desirable for multiple datasets visualization. In case of voxelized clipping, we see similar behavior. The rendering time increases considerably after 4 datasets. After that we see the similar trend of increasing rendering time with increasing *N*. The voxelized clipping object can be manipulated in real time by changing the texture coordinates. Finally, for isosurface extraction the rendering time is pretty much similar to that for volume rendering. We can see a sudden jump in rendering time after 5 datasets and then there is a rapid increase in the rendering time. Isosurface can be changed in real time; since this technique does not generate the polygonal representation of the isosurface.

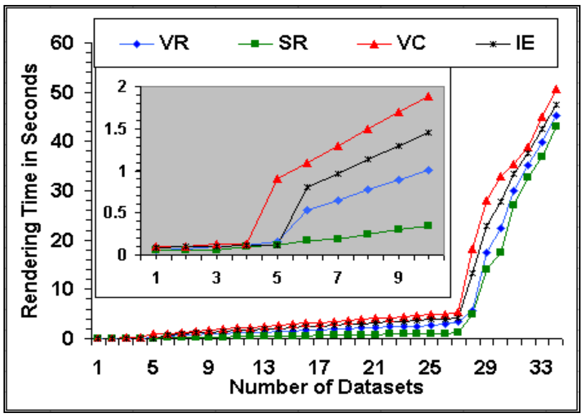


Figure 6: Rendering time for various techniques without volume shading. The inset shows the first transition in the low N regime. (VR- Volume Rendering, SR – Surface Rendering, VC- Voxelized Clipping, IE- Isosurface Extraction)

Considering the general trend in all the techniques Figure 6 without shading, we can see that rendering time shows two transitions; one around 5 dataset and another around 25 datasets. The first transition is due to the limited texture memory, which results in texture swapping. The second transition arises from the swapping between the main memory and the virtual memory. Since the second transition is due to the virtual memory, the rendering time increases much more rapidly after the second transition as compared to the increase after the first transition. In case of box clipping, such a high frame rate can be attributed to the fact that the number of slices being mapped is considerably less than the number of slices mapped in the case of actual volume rendering. With voxelized clipping approach, the frame rates are comparatively smaller due to the extra textures required for clipping. The first transition in rendering time occurs little early as compared to other techniques because of the extra texture.

4. VOLUME SHADING

Here we first give the brief description about he shading model we implemented and then do the performance analysis of the techniques mentioned in section 3 (volume rendering, clipping and isosurface extraction) when shading is enabled.

Shading Model

Illuminating the region of the interest in the data further facilitates the understanding. Volume shading adds the illumination term to the volume rendering. We have utilized the phong model of illumination for the clipped surface and isosurfaces. According to the phong model [Pho75], light at any given point is combination of diffuse, specular, and ambient component. These three components are added to determine the illumination or color of a point or polygon. Calculating the surface normal is done using the gradient. For each 3D texture a gradient texture corresponding to the scalar dataset is created which hold the normal for each voxel depending upon the scalar values. Therefore, for each 3D texture being visualized there is a gradient texture. Both these are passed to the GPU and lighting calculations are performed. This technique requires an additional texture per dataset, thus reducing the number of datasets that can be visualized without shading to half. Since we already have a pre-computed clipping texture, applying to the isosurface or the clipped volume is straightforward.

Rendering Time with Volume Shading

When shading is enabled, the maximum number of datasets that can be visualized together is reduced roughly to half. This is due to the requirement of one gradient texture per dataset. Gradient calculation can also be done but it would require 6 extra texture fetches for each voxel. Figure 7 shows the rendering time for the above techniques when the shading is enabled. Although decrease of rendering time with shading is primarily due to extra gradient texture, it is also related to the lighting calculation done by GPU for every visible fragment. The effect of shading can be clearly seen for all the four cases as the number of datasets for the transitions decreases and rendering time increases (Figure 7). The first transition occurs after 2 datasets, since while we are displaying 2 textures with shading; we are effectively dealing with 4 textures in the memory. The second transition occurs at 13 dataset in contrast to 27 datasets when shading is disabled. The frame rates are also affected in a similar way. For 4 datasets, volume rendering frame rate is around 9 fps with shading while it is 0.67 fps with shading. Similar behavior can be seen when MDV for 9 or 16 datasets is done. For 9 datasets with voxelized clipping the frame rates are 0.47 and 0.10 with and without shading respectively. Since the maximum number of datasets that can be handled together is only 17.

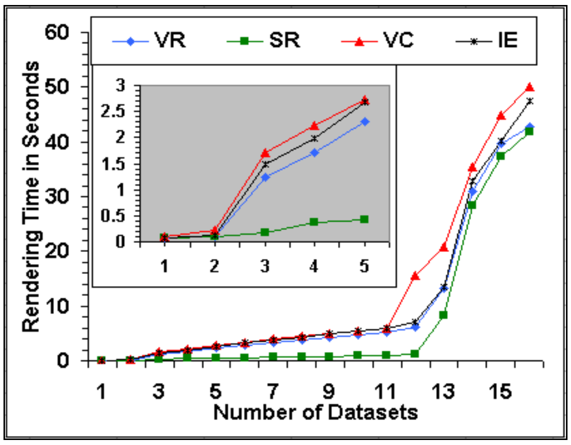


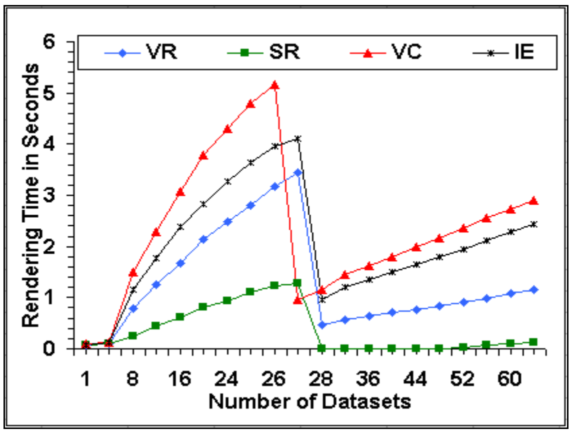
Figure 7: Rendering time for various techniques with shading. The inset shows the first transition in the low *N* regime. (VR- Volume Rendering, SR – Surface Rendering, VC- Voxelized Clipping, IE-Isosurface Extraction)

5. DYNAMIC RESOLUTION (DR) FOR IMPROVED MDV

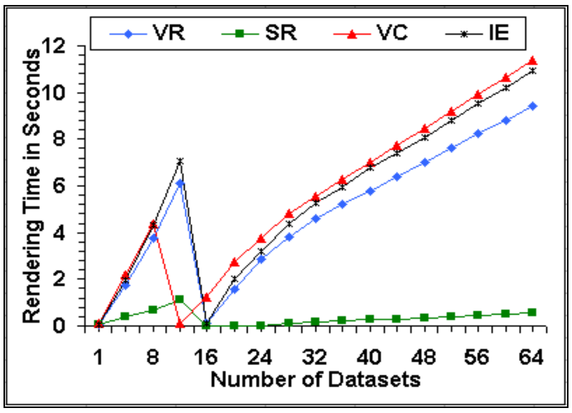
All the above texture based techniques (with and without shading) show the general trend that rendering time increases non-linearly as number of datasets (*N*) increase. The average slope of the *time-N* curve in the *high-N* region tends to be larger than that in the *low-N* region. An improved MDV method is expected to reduce the time in overall, maintain a linear *time-N* relationship over the entire *N-regime*, and increase N_{MAX}. The dynamic resolution approach is similar to the one proposed in [Kha06a]. The dynamic-resolution (DR) method, switches from a high- to a low-resolution mode as *N* increases. The basis for texture generation at a reduced resolution is the fact that once the dataset becomes too large, individual texels tends to be sub-pixel. This is also true in the case of MDV since with increasing *N*, individual viewports corresponding to individual datasets become smaller for a given window display size. Our dynamic-resolution (DR) approach sub-samples the volume data for a reduced resolution according to some criteria. For sub-sampling we use the octree data structure which sub samples the original volume in to eight sub volumes. Dynamic resolution checks for the user-defined threshold limits to decide whether the resolution needs to be reduced or not, every time a new texture is added. If increasing N does not break the threshold limit, more textures can be added at the current resolution. If the threshold limit is crossed, then the resolution of all the textures including the new ones is decreased. Reducing the resolution, frees up large amount of memory and thus more datasets can be visualized subsequently. Note that once the resolution is lowered, if the higher resolution is needed, the corresponding data have to be loaded again.

Improved MDV Results Using DR

Using the above-proposed DR approach, threshold criteria is defined to maintain the linear behavior of rendering time with increasing number of datasets (N).



(a)



(b)

Figure 8: (a) Rendering time for various techniques without shading with DR. (b) Rendering time for various techniques with shading with DR. (VR- Volume Rendering, SR – Surface Rendering, VC- Voxelized Clipping, IE- Isosurface Extraction)

The criteria for switching to lower resolution can be user defined. It can be based in rendering time, level of interactivity (number of frame rates) desired by the user, number of datasets or any other user defined criteria. The criterion avoids the relatively rapid increase in the rendering times by examining the *time-N* slope. For instance,

when, $R = \frac{T_{N+1} - T_N}{T_N - T_{N-1}} > 1.5$, we switch to a lower resolution mode for the *N+1* and more datasets. Here R, defines the slope, and T_N defines the rendering time in seconds for the N number of datasets. The above criteria, reduces the resolution when the second transition occurs. The first transition does not satisfy the above criteria and is thus ignored. The condition could be modified to capture the first transition and reducing the resolution. When shading is disabled the resolution is decreased around 27 datasets (Figure 8a). After this point we see the linear behavior till 64 datasets. Similarly, when shading is enabled, transition to the lower resolution occurs around 12th dataset (Figure 8b). For voxelized clipping the transition to lower resolution occurs little early due to the use of extra texture. Reducing the resolution also frees up memory and thus allowing more number of datasets to be loaded. Using the different criteria linear *time-N* slope can be obtained for all the three regions of the curve shown in Figure 6 and Figure 7.

6. CONCLUSIONS

In this paper, we have presented various texture-based multiple datasets visualization (MDV) techniques. With volume rendering, clipping and isosurface extraction the trend is similar in all the cases. In all the cases, we see two transitions in rendering time. First is due to swapping between main memory and texture memory. After second transition swapping of the texture objects takes place between the main memory and texture memory which dominates the rendering time. When volume shading is enabled an extra texture is needed to pass the surface normal to the GPU which reduces the overall number of textures that can be visualized simultaneously. To improve the performance we exploit the Dynamic-Resolution approach. The DR reduces the storage space so that more datasets can be loaded and a linear time-N slope can be maintained.

7. REFERENCES

[Cab94] Cabral, B., Cam, N. and Foran, J. Accelerated volume rendering and tomographic reconstruction using texture-mapping hardware. Proc. Symp. Volume Visualization. 91-98, 1994.

[Chi97] Chi, Ed Huai-hsin, Konstan, J., Barry P, and Riedl, J. A spreadsheet approach to information visualization. Proc. of the 10th annual ACM symposium on User interface software and technology, pp. 79-80, 1997.

[Cli88] Cline, H., Lorensen, W., Ludke S., Crawford, C., and Teter, B. Two algorithms for three dimensional reconstruction of tomographs, Medical Physics. Vol. 15, pp. 320-327, 1988.

[Cod05] Codes. Electronic calculation packages: PWScf (www.pwscf.org) and VASP (cms.mpi.univie.ac.at/vasp), 2005.

[Cru96] Crutcher, R.M., Baker, M.P., Baxter, H., Pixton, J. and Ravlin, H. Astronomical Data Analysis Software and Systems V, ASP Conference Series, Vol. 101., 1996.

[Cul93] Cullip, T. J. and Newman, U. Accelerating volume reconstruction with 3D texture hardware. Technical Report TR93-027, University of North Carolina, Chapel Hill, N. C., 1993.

[For92] Forguson. Visual Kinematics Inc, (Mountain View, CA USA), FOCUS User Manual, Release 1.2., 1992.

[Gel96] Van Gelder, A., and Kim, K. Direct volume rendering with shading via three-dimensional textures. Proc. Symp. Volume Visualization, pp. 23-30, 1996.

[Hau96] Haubner, M., Krapichler, Ch., Lösch, Englmeier, K. H and Van Eimeren W. Virtual Reality in Medicine – Computer Graphics and Interaction Techniques. IEEE Transactions on Information Technology and Medicine, Vol 1, pp 61-72, 1996.

[Geo06] Georgii J, Westermann R. A Generic and Scalable Pipeline for GPU Tetrahedral Grid Rendering, IEEE Transactions on Visualization and Computer Graphics, Vol.12, No.5, Sept./Oct. 2006.

[Had05] Hadwiger M., Sigg C., Scharsach H., Buhler K. and Gross M., Real-time ray-casting and advanced shading of discrete isosurfaces. Eurographics, 2005.

[Jan00] Jankun-Kelly, T. J., Ma, K. A spreadsheet interface for visualization exploration IEEE Proc. of the conference on Visualization, 2000.

[Kar06] Karki, B B., and Khanduja G. Computer Simulation and Visualization of Vacancy Defects in $MgSiO_3$ pervoskite. MSMSE, 2006.

[Kha05] Khanduja, G. and Karki, B. B. Visualization of 3D scientific datasets based on interactive clipping. WSCG, ISBN 80-903100-9-5., 2005.

[Kha06a] Khanduja, G. and Karki, B. B. Multiple Dataset Visualization using Isosurface extraction. VIIP, pp. 541-547, 2006.

[Kha06b] Khanduja, G. and Karki, B. B. A Systematic Approach to the Multiple Dataset Visualization of Scalar Volume Data. GRAPP, pp. 59-66, 2006.

[Kip05] Kipfer P. and Westermann R. GPU construction and transparent rendering of iso-surfaces. Proceedings of Vision, Modeling and Visualization, 2005.

[Lev94] Levoy, M. Spreadsheets for Images, ACM Proceedings of the 21st annual conference on Computer graphics and interactive techniques, Vol 28,pp. 139—146,1994.

[Lor87] Lorensen, W.E. and Cline H.E. Marching Cubes: A high-resolution 3D surface reconstruction algorithm. Computer Graphics, Proc. of SIGGRAPH, 21, pp. 163-169, 1987.

[Lor93] Lorensen, W. E. Geometric clipping using Boolean textures Visualization, Proc., IEEE, 1993.

[Mei00] Meibner, M., Huang, J., Bartz, D., Mueller, K. and Crawfis, R. A practical evaluation of popular volume rendering algorithms, Proc. IEEE Symps. Volume Visualization, pp. 81-90, 2000.

[Par98] Parker S., Shirley, P., Livnat, Y. Hansen, C., and Sloan, P. Interactive ray tracing for isosurface rendering, Proc. Visualization, pp. 233-238., 1998.

[Pho75] Phong, Bui-Tuong. Illumination for Computer-Generated Pictures, Communications of the ACM, vol. 18, no. 3, pp. 311-317, 1975.

[Sch04] Schulze, J.P., and Forsberg, A.S. User-friendly volume data set exploration in the Cave, http://www.calit2.net/~jschulze/publications/Schulze 2004b.pdf., 2004.

[Sch92] Schroeder, W., Zarge, J. A., and Loresen, W.E. Decimation of triangle meshes, Computer Graphics. SIGGRAPH Conf. Proc., pp. 65-70,1992.

[Wei00] Weiler, M., Westermann, R., Hansen, C., Zimmerman, K., and Ertl, T. Level-of-detail volume rendering via 3D textures. Proceedings of IEEE Volume Visualization, pp 7-13, 2000.

[Wei03] Weiskopf D., Engel, K., and Ertl, T. Interactive Clipping Techniques for Texture-Based Volume Visualization and Volume Shading. IEEE Transactions on Visualization and Computer Graphic, 9, 298-312, 2003.

[Wes98] Westermann, R. and Ertl, T. Efficiently using graphics hardware in Volume Rendering Applications. SIGGRAPH Conf. Proc., pp. 169-179, 1998.

[Wil02] Wilson B., Ma, K. and McCormick, P. S. A hardware-assisted hybrid rendering technique for interactive volume visualization. Proc. of Volume Visualization and Graphics Symposium, Vol 9, Issue 2, pp 123-130, 2002.

[Wil94] Wilson, O., Van Gelder, A. and Wilhems, J. Direct volume rendering via 3d textures. Technical Report UCSC-CRL-94-19, Jack Baskin School of Eng., Univ of California at Santa Cruz., 1994.