Evaluation of Mesh Compression and GPU Ray Casting for Tree Based AMR data in VTK

Antoine Roche[1,2,3] and Jérôme Dubois[1,3]

[1]Université Paris-Saclay, CEA, Laboratoire en Informatique Haute Performance pour le Calcul et la simulation, 91680 Bruyères-le-Châtel, France
[2]MSc Student, Université de Reims Champagne-Ardennes, France
[3]CEA, DAM, DIF, F-91297 Arpajon, France

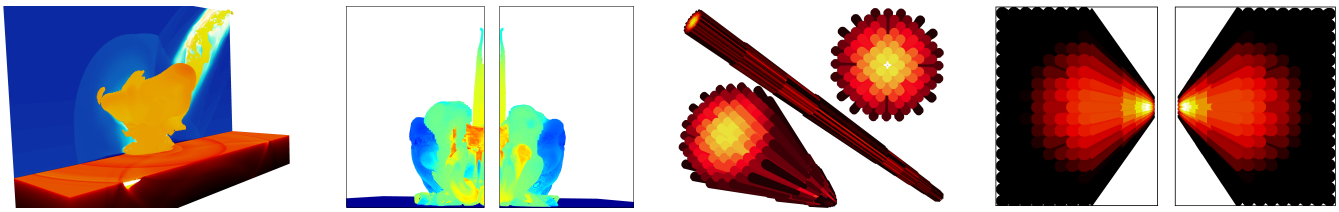

Figure 1: *Native Tree Based AMR volume data sets filtered with the Visualization ToolKit (VTK), converted as a compressed octree (SVDAG) and rendered on a NVidia GPU. First two left pictures show the asteroid at time step 25176, thresholded on water and basalt, with a vertical AxisCut displaying sound speed for first one. Pressure is displayed on first and temperature on second. Two most right pictures represent a fuel assembly from a nuclear reactor core, with a threshold on elements of high power for the first one, and the full raw assembly for the second. Split images show rendering with VTK on left half and SVDAG on right half. With our approach we could reduce memory footprint by ratios of 2 to 7x, loading time by 50 to 110x and size on storage by 5 to 45x.*

Abstract

Adaptive Mesh Refinement (AMR) methods are common in scientific workloads, and very useful during analyses and visualization. In this work, we aim to reduce memory and storage footprint of AMR data. We adapt tree compression (SVDAG) and GPU Ray Cast rendering, created for Computer Graphics surface scenes, for volume data from Scientific Visualization workloads. In particular, experiments have been conducted with the native Tree-Based AMR (TB-AMR) data structure in the Visualization ToolKit (VTK): vtkHyperTreeGrid (HTG). A HTG to SVDAG online converter has been implemented as well as a multi-SVDAG extension. Results showed several orders of magnitude memory footprint reduction thanks to the compression and efficient Ray Cast rendering. Furthermore, serialization to SVDAG enabled almost instant loading and display of simulation data instead of minutes. Overall our experiments show great benefits, and this shows great promises to further improve TB-AMR analyses.

CCS Concepts

• ***General and reference*** → ***Design; Performance;*** *Evaluation;* • ***Human-centered computing*** → ***Visualization;*** • ***Software and its engineering*** → *Designing software;* • ***Information systems*** → *Data compression;*

1. Introduction

Adaptive mesh refinement techniques are very useful to describe scenes with a good balance between accuracy, memory occupancy and computations. They can be used as the main support to describe a scene or simulation, or even used as an accelerating structure for specific applications such as isosurface generation [WG00] or accelerating the rendering of shadows [SKOA14]. We particularly take interest in Tree-Based Adaptive Mesh Refinement (TB-AMR) [BO84] techniques to handle massive high fidelity data sets in Scientific Visualization (SciVis). In SciVis one of the challenges is to be able to cope with large meshes, multiple properties and complex filtering as interactively as possible in a constrained memory and compute budget. In this aspect, the VTK HyperTreeGrid (HTG) data structure [HLPD19, DHL18], available in the Visual Toolkit (VTK), is the reference for TB-AMR data. Computer graphics (CG) community has also been developing techniques to enable visualization of large complex scenes. Shared with SciVis AMR techniques is the octree [Mea82, LK11] data structure, heavily used for 3D scenes representation. In some cases, specific op-

DOI: 10.2312/eurp.20201115

timizations can be introduced to be extremely efficient for limited applications such as surface rendering of scenes. Sparse Voxel Directed Acyclic Graph (SVDAG) [KSA13] and its derivatives like the Symmetry-Aware SVDAG [VMG16] are an example of such CG data structures, introducing lossless compression of octrees and enabling very interesting applications like the ultra-efficient surface representation of 3D-acquired scenes [KRB*16]. The efficiency of the proposed SVDAG lossless compression with optimal attribute management [DKB*16] and the provided GPU SVDAG ray tracer show great potential for enhancing VTK HyperTreeGrid for SciVis usage. In this paper we will first describe our main contributions, a conversion from HTG to SVDAG and the introduction of the multi-SVDAG, a per-level SVDAG structure, and then show the results achieved in the next section, before we conclude on the benefits of the combined techniques of HTG and SVDAG for SciVis purposes.

2. Implementations and contribution

To explore the potential of SVDAG techniques for TB-AMR meshes, we created several HTG to SVDAG converters and tested a multi-SVDAG approach to cope with leaf nodes of different levels.

HTG to SVDAG conversion. The SVDAG code we were granted access to by the authors of [DSKA18] builds a single SVDAG from a Morton order [ZPY*18] of the finest voxels, which are on the same level, and uses one 32bit color array ordered accordingly. More complex or efficient SVDAG implementations exist, and this implementation is sufficient for preliminary evaluation.

HTG is a grid of trees, so we create one SVDAG per tree, and use the merging process of [KSA13] to add abstract levels of trees, resulting in less but deeper SVDAGs. This process is then iterated until a single SVDAG remains. To be noted, larger leaf cells are oversampled to the finest level, and are then compressed during the SVDAG construction, greatly compensating the oversampling. However, extra attribute values are generated to match the extra nodes and these are not compressed, potentially resulting in extreme overhead for meshes with deep trees as we will see in the results section. Finally, we convert the selected HTG attribute to a color array with the VTK lookup table and palette, and use it for SVDAG rendering. Therefore, SVDAG and HTG colors are identical. Selecting another attribute would only require updating the color array, and so no re-computation of the SVDAG mesh would be needed. With this rendering, we avoid the generation of the costly unstructured surface representation of the filtered HTG, but need to generate the SVDAG once per scene.

Multi-SVDAG. To address the large amount of oversampled leaf cells, we generate one SVDAG for each level of leaf cells. Each SVDAG would hold the parent tree structure needed to represent leaf cells of one level or potentially a range of levels, thus limiting or canceling the impact of oversampling. As we will see in the results section, with this approach we call multi-SVDAG, the attribute array size is on par with the original HTG's, while showing high level of compression for the trees.

3. Results

We use the following SciVis data: FuelAssembly (FA) and Asteroid (YA31), illustrated in Figure 1. FA is a single fuel assembly from an international benchmark [HMP10] and YA31 is a HTG conversion of the asteroid fall simulation [PR17]. FA and YA31 are volume data, with respectively 222 and 311 million cells. Raw surface render of YA31 data is a rectangular box, and we apply VTK filtering before converting the resulting HTG to a SVDAG for our evaluation. Ultimately we can serialize the SVDAG using the cereal C++ library, allowing for ultra fast loading for pure visualization purposes, as no interactive filtering of the SVDAG has been implemented. SVDAG creation required about 30 seconds for each data set, and we could certainly optimize our process which was conducted serially on a laptop with a Intel i7 CPU, 32GB RAM, 16GB Nvidia M5000M GPU and 512GB SSD storage. In Table 1, we see the single SVDAG implementation cannot be computed for YA31 because of the extra memory cost of oversampling. Multi-SVDAG copes with this problem, and offers attribute array size on par with HTG's and several orders of magnitude less nodes than native HTG for both data sets. Fuel Assembly is a relatively dense data set, and we see the replication of parent nodes of our approach increases the amount of compressed nodes by a 5x factor compared to single SVDAG. Memory footprint is also greatly decreased, with ratios of 7.7x and 3.6x, because of two factors : efficient SVDAG tree compression and no generation of the unstructured surface mesh for visualization. Furthermore, loading time of a serialized SVDAG is several orders of magnitude faster partly due to size files 5 to 45x smaller. This performance and compression, combined with the attribute compression proposed in [DSKA18], would make SVDAG an export format to preview SciVis data in an approach comparable to Paraview Cinema [AJO*14], with full mesh and no reconstruction [LKA*18].

Table 1: *Comparing HTG, SVDAG and our multi-SVDAG. We give the overall (parents+leaves) count of nodes in the tree(s), size of the color attribute array, CPU+GPU memory footprint, size on storage and time needed to load from storage and render first image.*

FA	Nodes	Color	Mem.	Disk	L+R
HTG	222M	222M	6.5GB	25MB	25s
SVDAG	51K	202M	1.8GB	5MB	0.5s
m-SVDAG	276K	186M	1.8GB	5MB	0.5s

YA31	Nodes	Color	Mem.	Disk	L+R
HTG	311M	311M	14GB	460MB	110s
SVDAG	N/A	>5.7B	>23GB	>300MB	N/A
m-SVDAG	1.73M	186M	1.8GB	10MB	1s

4. Conclusion

We conclude that SVDAG compression applied to TB-AMR SciVis data is very promising. In the frame of VTK, applying such compression directly in the HTG data structure could provide several orders of magnitude compression, faster rendering and also reduce loading times. As future work, we are investigating NVidia Optix and Intel OSPRay volume rendering of the HTG, which could rely on an internal data structure similar to SVDAG, providing new advanced functionalities for the SciVis community.

References

[AJO*14] AHRENS J., JOURDAIN S., OLEARY P., PATCHETT J., ROGERS D. H., PETERSEN M.: An image-based approach to extreme scale in situ visualization and analysis. In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2014), pp. 424–434. 2

[BO84] BERGER M. J., OLIGER J.: Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics 53*, 3 (1984), 484 – 512. URL: http://www.sciencedirect.com/science/article/pii/0021999184900731, doi:https://doi.org/10.1016/0021-9991(84)90073-1. 1

[DHL18] DUBOIS J., HAREL G., LEKIEN J.-B.: Interactive visualization and analysis of high resolution hpc simulation data on a laptop with vtk. In *IEEE VIS 2018 Scientific Visualization Contest* (2018), IEEE VIS'18 SciVis Contest. 1

[DKB*16] DADO B., KOL T. R., BAUSZAT P., THIERY J.-M., EISEMANN E.: Geometry and attribute compression for voxel scenes. *Computer Graphics Forum 35*, 2 (2016), 397–407. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12841, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12841, doi:10.1111/cgf.12841. 2

[DSKA18] DOLONIUS D., SINTORN E., KÄMPE V., ASSARSSON U.: Compressing color data for voxelized surface geometry. *IEEE Transactions on Visualization and Computer Graphics 25*, 2 (2018), 1270–1282. doi:10.1109/TVCG.2017.2741480. 2

[HLPD19] HAREL G., LEKIEN J., PÉBAŸ P. P., DUBOIS J.: Efficient visualization and analysis of large-scale, tree-based, adaptive mesh refinement simulations with rectilinear geometry. *EWCO, tbp* (2019). 1

[HMP10] HOOGENBOOM J., MARTIN W., PETROVIC B.: Monte carlo performance benchmark for detailed power density calculation in a full size reactor core. 2

[KRB*16] KÄMPE V., RASMUSON S., BILLETER M., SINTORN E., ASSARSSON U.: Exploiting coherence in time-varying voxel data. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2016), ACM,. URL: exploiting_coherence_in_time-varying_voxel_data.pdfhttps://youtu.be/ptmHag3XwXY. 2

[KSA13] KÄMPE V., SINTORN E., ASSARSSON U.: High resolution sparse voxel dags. *ACM Trans. Graph. 32*, 4 (July 2013), 101:1–101:13. URL: http://doi.acm.org/10.1145/2461912.2462024, doi:10.1145/2461912.2462024. 2

[LK11] LAINE S., KARRAS T.: Efficient sparse voxel octrees. *IEEE Transactions on Visualization and Computer Graphics 17*, 8 (Aug 2011), 1048–1059. doi:10.1109/TVCG.2010.240. 1

[LKA*18] LUKASCZYK J., KINNER E., AHRENS J., LEITTE H., GARTH C.: Voidga: A view-approximation oriented image database generation approach. In *2018 IEEE 8th Symposium on Large Data Analysis and Visualization (LDAV)* (2018), pp. 12–22. 2

[Mea82] MEAGHER D.: Geometric modeling using octree encoding. *Computer Graphics and Image Processing 19*, 2 (1982), 129 – 147. URL: http://www.sciencedirect.com/science/article/pii/0146664X82901046, doi:https://doi.org/10.1016/0146-664X(82)90104-6. 1

[PR17] PATCHETT J. M., ROSS G.: Deep water impact ensemble data set. URL: https://sciviscontest2018.org/wp-content/uploads/sites/19/2017/09/DeepWaterImpactEnsembleDataSet_Revision1.pdf. 2

[SKOA14] SINTORN E., KÄMPE V., OLSSON O., ASSARSSON U.: Compact precomputed voxelized shadows. *ACM Transactions on Graphics 33*, 4 (2014). 1

[VMG16] VILLANUEVA A. J., MARTON F., GOBBETTI E.: Ssvdags: Symmetry-aware sparse voxel dags. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2016), I3D '16, ACM, pp. 7–14. URL: http://doi.acm.org/10.1145/2856400.2856420, doi:10.1145/2856400.2856420. 2

[WG00] WILHELMS J., GELDER A. V.: Octrees for faster isosurface generation. *IEEE TRANSACTIONS ON MEDICAL IMAGING 19* (2000), 739–758. 1

[ZPY*18] ZHOU G., PAN Q., YUE T., WANG Q., SHA H., HUANG S., LIU X.: Vector and raster data storage based on morton code. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLII-3* (2018), 2523–2526. URL: https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-3/2523/2018/, doi:10.5194/isprs-archives-XLII-3-2523-2018. 2