


# Volumetric Rendering on Wavelet-Based Adaptive Grid

Alexei V. Vezolainen <sup>1,\*</sup>, Gordon Erlebacher <sup>2</sup>, Oleg V. Vasilyev <sup>3</sup>  and David A. Yuen <sup>4</sup><sup>1</sup> NPO Geoved LLC, 196644 Saint Petersburg, Russia<sup>2</sup> Department of Scientific Computing, Florida State University, Tallahassee, FL 32306, USA; gerlebacher@fsu.edu<sup>3</sup> Keldysh Institute of Applied Mathematics of Russian Academy of Sciences, 125047 Moscow, Russia; oleg.v.vasilyev@gmail.com<sup>4</sup> Department of Applied Physics and Applied Mathematics, Columbia University, New York, NY 10027, USA; daveyuen@gmail.com

\* Correspondence: a\_vez@mail.ru

**Abstract:** Numerical modeling of physical phenomena frequently involves processes across a wide range of spatial and temporal scales. In the last two decades, the advancements in wavelet-based numerical methodologies to solve partial differential equations, combined with the unique properties of wavelet analysis to resolve localized structures of the solution on dynamically adaptive computational meshes, make it feasible to perform large-scale numerical simulations of a variety of physical systems on a dynamically adaptive computational mesh that changes both in space and time. Volumetric visualization of the solution is an essential part of scientific computing, yet the existing volumetric visualization techniques do not take full advantage of multi-resolution wavelet analysis and are not fully tailored for visualization of a compressed solution on the wavelet-based adaptive computational mesh. Our objective is to explore the alternatives for the visualization of time-dependent data on space-time varying adaptive mesh using volume rendering while capitalizing on the available sparse data representation. Two alternative formulations are explored. The first one is based on volumetric ray casting of multi-scale datasets in wavelet space. Rather than working with the wavelets at the finest possible resolution, a partial inverse wavelet transform is performed as a preprocessing step to obtain scaling functions on a uniform grid at a user-prescribed resolution. As a result, a solution in physical space is represented by a superposition of scaling functions on a coarse regular grid and wavelets on an adaptive mesh. An efficient and accurate ray casting algorithm is based just on these coarse scaling functions. Additional details are added during the ray tracing by taking an appropriate number of wavelets into account based on support overlap with the interpolation point, wavelet coefficient magnitude, and other characteristics, such as opacity accumulation (front to back ordering) and deviation from frontal viewing direction. The second approach is based on complementing of wavelet-based adaptive mesh to the traditional Adaptive Mesh Refinement (AMR) mesh. Both algorithms are illustrated and compared to the existing volume visualization software for Rayleigh-Benard thermal convection and electron density data sets in terms of rendering time and visual quality for different data compression of both wavelet-based and AMR adaptive meshes.

**Keywords:** adaptive mesh; volumetric rendering; volume rendering; wavelet compression; scientific visualization



**Citation:** Vezolainen, A.V.; Erlebacher, G.; Vasilyev, O.V.; Yuen, D.A. Volumetric Rendering on Wavelet-Based Adaptive Grid. *Fluids* **2022**, *7*, 245. <https://doi.org/10.3390/fluids7070245>

Academic Editor: Mehrdad Massoudi

Received: 14 June 2022

Accepted: 11 July 2022

Published: 16 July 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Numerical modeling of physical phenomena frequently involves processes across a wide range of spatial and temporal scales. To ensure better capturing of the flow physics on a near optimal adaptive computational mesh with substantially smaller computational cost, while resolving the localized structures of the solution, a number of wavelet-based adaptive computational methods have been developed recently (e.g., [1–4]).

Similarly to adaptive mesh refinement (AMR) methods (e.g., [5–7]), adaptive wavelet-based methods [3,8] provide a convenient and efficient approach to resolving multi-scale

processes on dynamically adaptive computational mesh, which would be hard-pressed to be solved by conventional finite-difference techniques on a uniform non-adaptive grid. Wavelet-based methods take advantage of the wavelet compression properties, as a result, functions with localized regions of sharp transition are well compressed using wavelet decomposition briefly discussed in Section 2. The adaptation is achieved by retaining only those wavelets, whose coefficients are greater than a given wavelet threshold (see Equation (2)). Thus, high-resolution computations are carried out only in those regions, where sharp transitions occur. In addition to controlling global  $L_2$  approximation error, wavelet based adaptive grid provides a compression factor of  $10\text{--}10^2$  relative to the uniform grid of the same effective resolution.

Since all the computations are performed in wavelet space on the adaptive grid, it makes sense to utilize the available data compression and perform data visualization in wavelet space on the same adaptive grid. Despite the wide use of AMR methods in high performance computing (e.g., [9,10]) and fast growing development of adaptive wavelet-based methods (e.g., [3,4,11–14]) visualizing data on adaptive grids has only limited level of support in widely used visualization packages such as VTK [15], ParaView [16], VisIt [17], and ChomboVis [18]. Rendering such data on adaptive meshes interactively with high visual quality, i.e., without holes and artifacts, still constitutes a challenge [19].

One of the first adaptive mesh volume rendering systems based on cell projection volume rendering was introduced by Ma and Crockett [20]. Over the years, a number of approaches for volume rendering of data on AMR meshes have been developed. For example, a slice-based volume rendering of 3D textures [21] and its multi-pass [22] and single-pass [23] extensions were developed for adaptive mesh overlapping coarse and fine-resolution AMR levels. An efficient method for rendering AMR data-based inter-block interpolation was developed in [24]. Examples of multi-resolution volume rendering methods for adaptive meshes based on the submission of lower-resolution hexahedral cells and local interpolants can be found in [25].

Despite the development of many successful volume-rendering methods for AMR data and the constantly growing need for widely available AMR data visualization tools, these methods have not been implemented in existing visualization packages. We present here several techniques for volume rendering of multi-scale data sets while capitalizing on the available sparse data representation. The first visualization technique is based on the direct summation of wavelet coefficients during volumetric ray casting. The second approach is to complement our adaptive wavelet grid with the appropriate number of nodes in order to produce a traditional AMR grid. Volume rendering on an AMR grid is performed either with the existing or with our own software. Our work has some parallels with a recently developed non-wavelet-based volume rendering method for representing and traversing adaptive octree data [19], where, similarly to the present paper, it was also demonstrated that the developed approach could be adopted for production use in ParaView [16].

## 2. Wavelet Based Solution of PDEs

The use of second generation wavelets [26] for the solution of partial differential equations has been extensively discussed in [4,27,28]. Partial differential equations are discretized at collocation points thus leading to a system of algebraic equations for the unknown wavelet coefficients at these collocation points. Grid adaptation is obtained by keeping the points with wavelet coefficients (from a previous time step) larger than a predefined threshold limit.

As a result of the wavelet transform, a function in physical space  $f(\mathbf{x})$  is represented by a superposition of scaling functions  $\phi_{\mathbf{l}}^0(\mathbf{x})$  ( $\mathbf{l} \in \mathcal{L}^0$ ), at the coarsest level of resolution, and wavelets  $\psi_{\mathbf{k}}^{\mu,j}(\mathbf{x})$  ( $\mathbf{k} \in \mathcal{K}^{\mu,j}$ ) of different families ( $\mu = 1, \dots, 2^n - 1$ ) and levels of resolution ( $j = 1, \dots, j_{\max}$ ) on an adaptive mesh:

$$u(\mathbf{x}) = \sum_{\mathbf{l} \in \mathcal{L}^0} c_{\mathbf{l}}^0 \phi_{\mathbf{l}}^0(\mathbf{x}) + \sum_{j=1}^{j_{\max}} \sum_{\mu=1}^{2^n-1} \sum_{\mathbf{k} \in \mathcal{K}^{\mu,j}} d_{\mathbf{k}}^{\mu,j} \psi_{\mathbf{k}}^{\mu,j}(\mathbf{x}), \tag{1}$$

where  $n$  is the dimensionality of the space, the bold subscripts  $\mathbf{l}$  and  $\mathbf{k}$  denote  $n$ -dimensional indices, while  $\mathcal{L}^0$  and  $\mathcal{K}^{\mu,j}$  stand for the associated index sets, and  $J$  is the maximum level of resolution that is present or allowed in the wavelet approximation. The scaling functions  $\phi_{\mathbf{l}}^0$  and the wavelets  $\psi_{\mathbf{k}}^{\mu,j}$  are constructed on a set of nested tensorial meshes with one-to-one correspondence between grid points and functions. The coefficients  $c_{\mathbf{l}}^0$  and  $d_{\mathbf{k}}^{\mu,j}$  represent, respectively, the averaged values and the local variation details of the field  $f(\mathbf{x})$  at different scales. The value of  $f(\mathbf{x})$  in physical space can be obtained either through an inverse wavelet transform to a regular grid of required resolution or through a direct application of Equation (1) at the required point of physical space. Scaling and wavelet functions normally have local support (see one-dimensional scaling function  $\phi(\xi)$  and wavelet  $\psi(\xi)$  Figure 1), which significantly limits the number of coefficients to sum in the Equation (1).

Wavelet-compressed solution  $\bar{f}^{>\epsilon}(\mathbf{x})$  on an adaptive mesh is obtained in the wavelet space by using wavelet coefficient thresholding. Namely, the wavelet filtered function is defined by

$$\bar{u}_i^{>\epsilon}(\mathbf{x}) = [\overline{u_i(\mathbf{x})}]^{>\epsilon} = \sum_{\mathbf{l} \in \mathcal{L}^0} c_{\mathbf{l}}^0 \phi_{\mathbf{l}}^0(\mathbf{x}) + \sum_{j=0}^{j_{\max}} \sum_{\mu=1}^{2^n-1} \sum_{\substack{\mathbf{k} \in \mathcal{K}^{\mu,j} \\ |d_{\mathbf{k}}^{\mu,j}| > \epsilon \|u\|}} d_{\mathbf{k}}^{\mu,j} \psi_{\mathbf{k}}^{\mu,j}(\mathbf{x}), \tag{2}$$

where  $\epsilon > 0$  stands for the non-dimensional (relative) threshold value,  $\|f\|$  being the (absolute) dimensional scale of  $f$ . For the details of the adaptive wavelet collocation method, the reader is referred to Refs. [4,27,28]. Note that a substantially larger threshold value  $\epsilon$  can be used for visualization purposes compared to the one used in computations. Thus, an additional level of compression is possible if the data are further compressed with larger  $\epsilon$ , which would further improve the efficiency of the visualization.

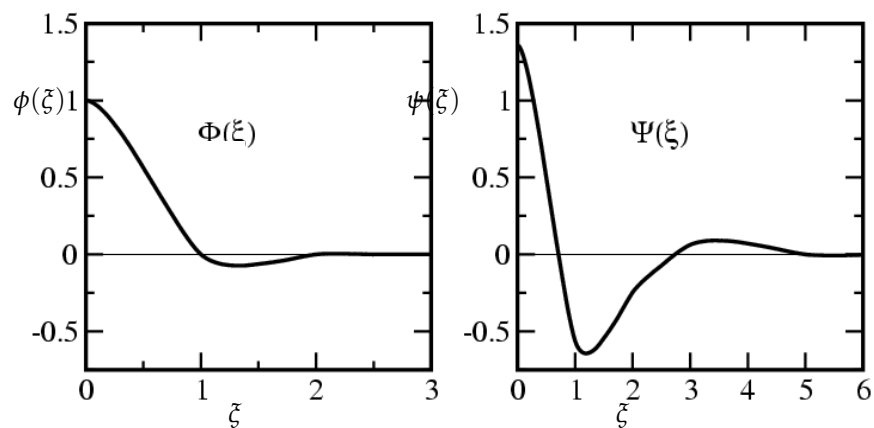


Figure 1. A 4-th order interpolating scaling function  $\phi(\xi)$  and wavelet  $\psi(\xi)$  vs. non-dimensional coordinate  $\xi$ .

### 3. Volume Rendering in Compression Domain

Compression properties of wavelet transform have been widely used for the visualization of large data sets [29–33]. Regularly sampled data have been projected into a wavelet basis. Depending on the required image quality, a large number of wavelet coefficients were neglected, thus leading to a significant reduction of memory usage during a local rendering (or during a network transmission for the rendering on a remote client). Interactive time rendering has been implemented for X-ray-like images via wavelet splatting and Fourier projection slice theorem [34,35]. Another interactive rendering technique, a walk through

the large data set by Guthe et al. [32] has been performed with the help of a block-wise wavelet transform [36,37]. The relevant blocks of voxels have been decompressed on the fly and rendered using hardware texture mapping [32].

Ray tracing in wavelet domain has been performed in non-interactive time. A function value  $f(x)$  has been reconstructed from its wavelet decomposition and used for the computing of rendering integral [31]. Alternatively, separate wavelet decompositions have been used for the reconstruction of opacity and color values used in the computing of rendering integral [38].

Interactive frame rates have been obtained in a parallel ray casting implementation by using block-wise wavelet transform on a multiprocessor system [36]. Also, interactive rendering of large data sets (fastest along axes directions) has been obtained by using “thick rays” in a shear-warped “foveated volume” [39]. This approach resembles volume rendering on adaptive mesh refinement (AMR) grids.

Adaptive mesh refinement has been extensively used for the simulations of multi-scale physical phenomena. A number of visualization techniques have been developed directly for the adaptive meshes, without resampling the data on a uniform grid (e.g., [40–42]). Extrapolation of wavelet-based grid data into some additional nodes will generate a correspondent AMR grid suitable for the application of the existing adaptive mesh visualization algorithms and software.

#### 4. Numerical Results and Discussion

A number of open-source software for volume rendering is currently available. Mostly, the rendering is performed on a regular grid, the size of which is limited by the available machine memory, e.g., VolPack library [43,44]. Unstructured grid software is more appropriate for the purposes of wavelet-based data visualization. VTK, the visualization toolkit, provides a number of libraries capable of dealing with unstructured grids [45]. ParaView [16], a VTK-based parallel visualization application, can display data on an unstructured grid [16]. Additionally, VisIt [17] and ChomboVis [18] are capable of displaying volume data on unstructured adaptive mesh.

##### *Available Volume Rendering Software*

A number of open-source software for volume rendering is currently available. Mostly, the rendering is performed on a regular grid, the size of which is limited by the available machine memory, e.g., VolPack library [43,44]. Unstructured grid software is more appropriate for the purposes of wavelet-based data visualization. VTK, the visualization toolkit, provides a number of libraries capable of dealing with unstructured grids [45]. ParaView, a VTK-based parallel visualization application, can display data on an unstructured grid [16]. Additionally, VisIt [17] and ChomboVis [18] are capable of displaying volume data on unstructured adaptive mesh.

#### 5. The Proposed Rendering Techniques

Several techniques for volume rendering of multi-scale data sets are presented. The first approach is based on the direct summation of wavelet coefficients during volumetric ray casting.

##### *5.1. Direct Summation of Wavelets*

First of all, rather than working with a wavelet scaling function of the largest possible support at the initially defined coarse grid, we perform a partial inverse wavelet transform as a preprocessing step to obtain scaling functions on a coarse grid at a user-prescribed resolution ( $64^3$  seems to be an optimal one for the current implementation). This preprocessing step is performed in a fraction of a second, although it may require some additional memory to store new scaling coefficients.

Interactive time ray casting algorithm is based just on these scaling coefficients. Additional detail is added during the ray tracing by taking an appropriate number of wavelets

into account based on support overlap with the interpolation point and wavelet amplitude. Other characteristics, such as opacity accumulation (front to back ordering) and deviation from frontal viewing direction could also be taken into account.

### 5.2. Ray Casting

The main goal of ray casting is to evaluate the solution of radiative transfer equation:

$$I = \int_0^{\infty} C \alpha e^{-\int_0^s \alpha ds'} ds, \quad (3)$$

where color and opacity are determined by the function  $f(\mathbf{x})$  in physical space,  $C = C(f(\mathbf{x}))$  and  $\alpha = \alpha(f(\mathbf{x}))$ , respectively. In the current implementation we use the following approximation of the rendering integral:

$$I \approx \sum_{i=0}^{n(\alpha)} C_i \alpha_i d_i \prod_{j=0}^{i-1} (1 - \alpha_j d_j), \quad (4)$$

where  $i$  corresponds to the index of ray point,  $d_i$  is the current step along the ray, and  $n(\alpha)$  is the total number of points along the ray.

In our current implementation of the algorithm, the step along the ray  $d_i$  is constant, depending on the number of fine levels on the adaptive mesh. A coarse image (of  $64^3$  scaling coefficients) is obtained in an interactive time by linear interpolations from the nearest eight coarse mesh scaling coefficients into the correspondent points on a ray. Computing of additional details requires the summation of fine level wavelet coefficients, which is currently non-interactive. It should also be noted that no hardware acceleration is used in the current implementation.

### 5.3. Data Structure

A data structure has been built for the rendering process (Figure 2). A linear array of coarse mesh nodes has been created for the storage of the scaling coefficients (an array of size of  $64^3$  in the current implementation) (Figure 2). Each coarse mesh node keeps the values of 8 nearest scaling coefficients. It should be noted that such redundancy provides 30% or so acceleration versus a non-redundant array of the scaling coefficients due to a better cache coherence of the redundant structure.

```
CoarseMeshNode coarse_mesh_node [N*N*N];

struct CoarseMeshNode {
    float scaling_coefficients [8];
    WaveletNode* wavelet_node_array;
    int number_of_wlt_nodes_of_level [J-1];
};

struct WaveletNode {
    float wavelet_coefficient;
    float wavelet_coordinates [3];
    int wavelet_type;
};
```

**Figure 2.** Data structure used for volume rendering. Scaling coefficients are stored as  $N^3$  array of “Coarse Mesh Nodes” ( $N = 64$  in the current implementation). Wavelet coefficients inside a coarse mesh block are stored in a dynamically allocated array of “Wavelet Nodes”.

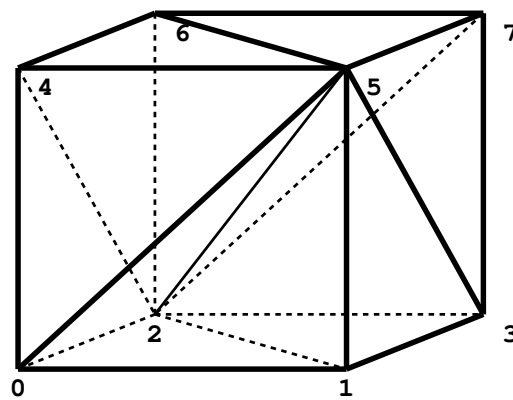
Each coarse mesh node has a pointer to a dynamically allocated adaptive array of wavelet coefficients located inside the correspondent coarse mesh block (“Wavelet Nodes” at Figure 2). The length of the array of wavelet coefficients is determined during the preprocessing step and is a constant for the rendering of time-independent data. The length





volume rendering. Additionally, the quality of volume rendering is low and seems to be inappropriate for our purposes of visualization. Thus, AMR grid data are to be written in an unstructured mesh VTK format.

Each cubic cell of the AMR grid is considered to be a combination of six tetrahedra (Figure 4). If we number all the nodes in the cell according to the numbering of VTK\_VOXEL cell type, six correspondent tetrahedra, or VTK\_TETRA cells, will be: 0125, 1325, 0245, 2645, 3725, 7625 [VTK, 2000]. It should be noted that such a triangulation is not compatible across the neighboring cells (i.e., T-junctions may appear) and therefore is potentially subject to various rendering artifacts, including gap (empty space) appearance during isosurface generation [46]. Note when a size of the input file is a concern one may save AMR data in a VTK\_VOXEL cell format instead of VTK\_TETRA cell format [VTK, 2000]. It might be convenient for slice or isosurface generation. As for the volume rendering, VTK seems to perform it on an unstructured tetrahedral mesh only; hence the volume is still to be tetrahedralized (and the additional memory for the tetrahedra to be requested).



**Figure 4.** Tetrahedral subdivision of cubic cell of AMR grid. The tetrahedra are: 0125, 1325, 0245, 2645, 3725, 7625.

## 6. Results

We have timed our volume rendering implementations for several data sets on a single processor system: Pentium 4, 3.2 GHz, 2G RAM, with Nvidia GeForce Fx5200-TD128 graphics card. Preprocessing time was not taken into account. The direct summation algorithm has been implemented through a C++ code and the rendering time was the time to compute the values of all the pixels of  $500 \times 500$  image matrix. VTK-based rendering has been performed through a Tcl/Tk script and the rendering time has been computed as the time of execution of `vtkRenderWindow` class command `Render`. We note that the rendering studies were performed on old generation system and the reported timings would be much shorter for systems with more modern processors.

### 6.1. Convection Data Sets

Constant viscosity  $256^3$  thermal convection data with Rayleigh number of  $10^{10}$  has been processed to simulate the output of a wavelet-based PDE solver. A 4-th order second generation wavelet transform [26] has been used for the data processing. The data has been assumed to be periodic in order not to deal with the distortions of wavelet functions near the boundaries.

The summary of the data set properties is shown in Table 1. Different thresholds for wavelet coefficients produce different data compressions (relative to the regular  $256^3$  grid). Completing the original wavelet-based grid to the AMR grid may significantly increase the number of grid nodes for the data set, especially if the original wavelet grid compression is relatively high.

**Table 1.** Data set properties and timings:  $\epsilon$  is the threshold,  $\sigma$  is the compression for the original wavelet based grid,  $\sigma_{amr}$  is the compression for the correspondent AMR grid,  $F$  and  $F_{vtk}$  are binary file sizes (in megabytes) to store the original wavelet and tetrahedralized AMR grid in VTK format, respectively.  $T_{wlt}$ ,  $T_{PT}$ ,  $T_{RC}$ , and  $T_{ZS}$  are the rendering times (in seconds) for the direct wavelet summation algorithm and for the VTK library unstructured grid renderers: ProjectedTetrahedra, RayCast, and ZSweep, respectively. A hyphen (-) implies the lack of RAM for the rendering.

Data Set	$\epsilon$	$\sigma$ , %	$\sigma_{amr}$ , %	$F$ , M	$F_{vtk}$ , M	$T_{wlt}$	$T_{PT}$	$T_{RC}$	$T_{ZS}$
Convection	0.0005	22.5	87.1	101	2199	254	-	-	-
	0.001	16.7	74.9	75	1847	198	-	-	-
	0.005	6.85	39.9	31	915	94	149	-	-
	0.01	4.51	27.3	21	607	70	96	-	-
	0.05	1.99	8.15	9.0	167	42	25	100	-
	0.1	1.72	3.82	7.8	83	41	10	58	-
	0.5	1.56	1.55	7.1	38	39	4.8	43	97
3D electron	$10^{-7}$	8.12	28.8	37	706	107	106	-	-
	$10^{-6}$	3.69	12.9	17	306	58	46	-	-
	$10^{-5}$	1.84	4.71	8.3	109	37	16	54	7400
	$10^{-4}$	1.21	1.65	5.5	41	32	5.2	35	210
	$10^{-3}$	0.83	1.04	3.8	26	31	3.6	29	78
	$10^{-2}$	0.41	0.60	1.9	15	29	2.1	25	57
	$10^{-1}$	0.10	0.22	0.5	5	19	0.7	22	38

### 6.2. Hydrogen 3D Orbital Data Sets

An analytical function, electron density for 3D hydrogen orbital, has been used to generate the data set on a regular  $256^3$  grid. A 4-th order interpolating wavelet transform has been used to compress the data to simulate the output of a wavelet-based PDE solver. The summary of the data set properties is shown in Table 1.

### 6.3. Rendering Results

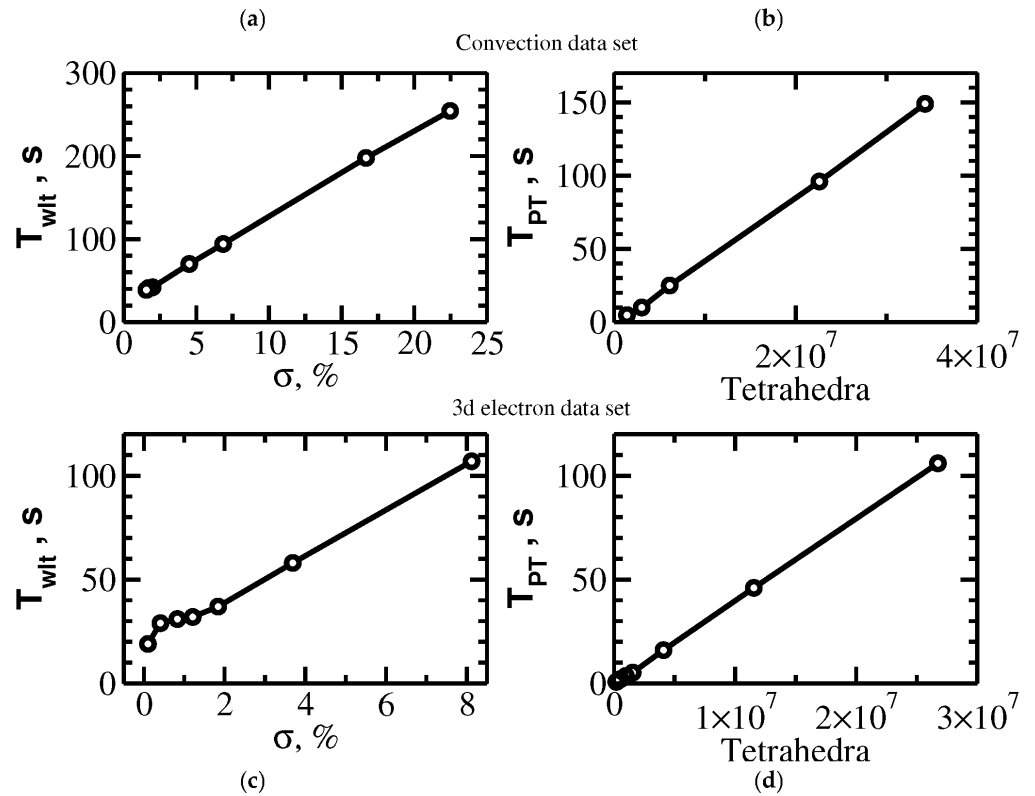
Rendering times for both data sets are shown in the Table 1 for the direct wavelet summation approach and for three VTK library unstructured grid renderers with VTK’s default settings. The projected tetrahedra renderer seems to provide fast volume rendering with adequate quality. Although the best rendering quality is provided by the much slower ZSweep renderer. It should be noted that VTK rendering with the default settings requires a lot of memory (Table 1) and therefore is not appropriate for large data sets. Apparently,  $\sim 5 \times 10^7$  is the maximum number of tetrahedra to be rendered on our system by VTK library.

VTK rendering through projected tetrahedra gives the rendering time linear to the number of tetrahedra in the unstructured mesh (Figure 5). In the direct wavelet summation approach, the rendering time is linear relative to the number of rays, the average number of points on a ray, and the number of wavelet coefficients in the sum of Equation (1) (Figure 5). Contrary to VTK, the direct summation approach does not require much memory and works successfully for all the data sets (Table 1). The rendering quality of direct wavelet summation on adaptive mesh is comparable to that of VTK’s renderers as well as to the quality of Amira [47], volume rendering on a regular  $256^3$  grid for both convection and hydrogen data sets (Figures 6 and 7). Noticeable color discrepancies in the rendering results are attributed to different ray integration approaches in the different renderers. Some rendering artifacts appear in the images produced by some VTK’s renderers (Figure 7).

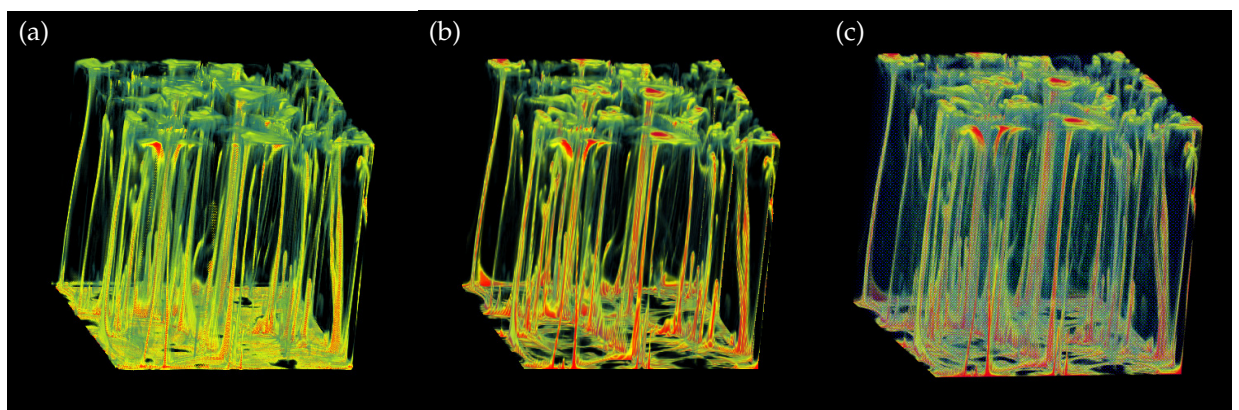
Increasing the threshold value  $\epsilon$  leads to a smaller number of wavelet coefficients being taken into account during the rendering. The error in the determining the function value  $f(x)$  through the truncated version of Equation (1) with all wavelet coefficients greater than the threshold is  $O(\epsilon)$  (e.g., [27]). For both data sets considered, the rendering results appear not to be changed for all the threshold values below 0.01. Therefore, we recommend to use that threshold value as an optimal one for a preprocessing step, before



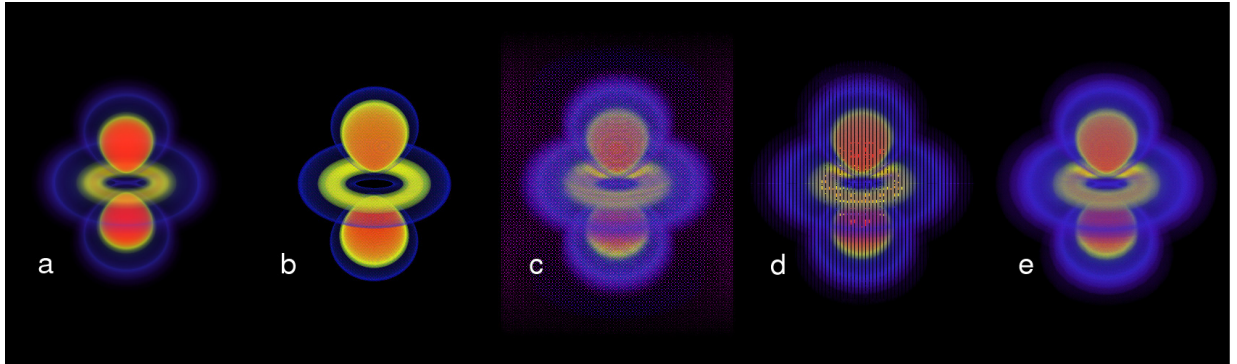
direct summation or tetrahedralization. The estimated errors in the rendering image,  $\sim 1\%$ , seem to be admissible for visualization purposes. Additionally, a larger threshold value strongly decreases the number of tetrahedra in the correspondent AMR grid hence making memory-consuming VTK-based visualization approach feasible.



**Figure 5.** Direct summation rendering time  $T_{wlt}$  vs. data compression  $\sigma$  for the convection (a) and electron density (c) data sets. The coarse mesh is  $64^3$ , the number of rays is  $500^2$ , the average number of points per ray is 170. The cutoff limits for the scaling and wavelet functions are 1 and 2, respectively. VTK projected tetrahedra rendering time  $T_{PT}$  vs. the number of tetrahedra in the correspondent AMR grid for for the Rayleigh–Benard convection (b) and electron density (d) data sets.



**Figure 6.** Convection data set. Direct wavelet summation adaptive mesh rendering (a) vs. Amira volume rendering (after extrapolation to a regular  $256^3$  grid) (b) vs. VTK’s projected tetrahedra volume rendering (after completing an AMR grid) (c). Wavelet compression is  $\sim 4.5\%$  relative to  $256^3$  regular mesh.



**Figure 7.** 3D electron density data set. Direct wavelet summation adaptive mesh rendering (a) vs. Amira volume rendering (after extrapolation to a regular  $256^3$  grid) (b) vs. VTK's projected tetrahedra (c), ray casting (d), and z-sweep (e) volume rendering. Wavelet compression is  $\sim 1.2\%$  relative to  $256^3$  regular mesh.

## 7. Conclusions and Future Work

Two visualization techniques have been presented for volume rendering on wavelet-based adaptive grids: direct summation of wavelet coefficients during volumetric ray casting, and complementing adaptive wavelet grid to a correspondent AMR grid for the subsequent VTK-based rendering. Both approaches provide relatively good image quality and reasonable timing. The direct summation approach provides a better image quality. In addition, it does not subject to memory restrictions, i.e., if the data set has been computed at some system using a wavelet-based PDE solver, the volume rendering can be performed on the same system. Although, “completing to an AMR” approach seems to be more general in the sense that it moves the problem of volume rendering on a wavelet-based grid into the more developed area of AMR grid visualization. Additionally, VTK provides a large number of tools to make the visualization process, if memory permits, more general. For memory saving, the number of nodes in a wavelet-based grid and therefore the number of tetrahedra in the corresponding AMR grid should be decreased. The obvious approach is to change thresholding. It is also possible to consider a different tetrahedralization technique than the one presented.

As for the summation of wavelet coefficients approach, alternative ways of ray integration could be considered. Particularly, the step along the ray is to be adaptive and the number of rays should not be fixed from the beginning of the rendering. Additionally, direct summation through Equation (1) could be changed to an exact wavelet transform into the corners of the block (of appropriate resolution) around the point of interest on a ray.

Hardware acceleration could be used for wavelet coefficient summation as well as for the coarse mesh interactive rendering. Additionally, coarse mesh interactive rendering could be performed through a faster algorithm, e.g., through shear-warp approach [43] It would allow interactive rendering on a regular grid larger than the current  $64^3$  grid, which could filter out several levels of wavelet coefficients to sum during the non-interactive phase of the rendering.

These alternative approaches to the volume rendering on wavelet-based adaptive grids are currently under consideration.

**Author Contributions:** Data curation, A.V.V., G.E. and D.A.Y.; investigation, A.V.V. and G.E.; methodology, A.V.V., G.E. and O.V.V.; resources, G.E. and D.A.Y.; supervision, G.E. and O.V.V.; validation, A.V.V.; visualization, A.V.V.; writing, original draft preparation, A.V.V. and O.V.V.; writing, review and editing, A.V.V. and O.V.V. All authors read and agreed to the published version of the manuscript.

**Funding:** D.A.Y. was partially supported by the DOE under grant DE-SC0019759 and by the NSF under grant EAR-1918126.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

AMR	adaptive mesh refinement
PDE	partial differential equation
VTK	Visualization Toolkit
VT VOXEL	orthogonal parallelepiped-type VTK cell
VTK TETRA	tetrahedra type VTK cell

## References

- Vasilyev, O.V.; Bowman, C. Second generation wavelet collocation method for the solution of partial differential equations. *J. Comput. Phys.* **2000**, *165*, 660–693. [\[CrossRef\]](#)
- Cruz, P.; Mendes, A.; Magalhães, F.D. Wavelet-based adaptive grid method for the resolution of nonlinear PDEs. *AIChE J.* **2002**, *48*, 774–785. [\[CrossRef\]](#)
- Schneider, K.; Vasilyev, O.V. Wavelet Methods in Computational Fluid Dynamics. *Ann. Rev. Fluid Mech.* **2010**, *42*, 473–503. [\[CrossRef\]](#)
- Nejadmalayeri, A.; Vezolainen, A.; Brown-Dymkoski, E.; Vasilyev, O.V. Parallel adaptive wavelet collocation method for PDEs. *J. Comput. Phys.* **2015**, *298*, 237–253. [\[CrossRef\]](#)
- Berger, M.J.; Olinger, J. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.* **1984**, *53*, 484–512. [\[CrossRef\]](#)
- Berger, M.; Colella, P. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.* **1989**, *82*, 64–84. [\[CrossRef\]](#)
- Linde, T.J.; Weirs, V.G.; Plewa, T. *Adaptive Mesh Refinement—Theory and Applications*; Springer: Berlin/Heidelberg, Germany, 2008.
- Vasilyev, O.V.; Yuen, D.A.; Paolucci, S. The Solution of PDEs Using Wavelets. *Comput. Phys.* **1997**, *11*, 429–435. [\[CrossRef\]](#)
- Kiris, C.C.; Barad, M.F.; Housman, J.A.; Sozer, E.; Brehm, C.; Moini-Yekta, S. The LAVA Computational Fluid Dynamics Solver. In Proceedings of the 52nd Aerospace Sciences Meeting, National Harbor, MD, USA, 13–17 January 2014. [\[CrossRef\]](#)
- Adams, M.; Colella, P.; Graves, D.; Johansen, H.; Keen, N.; Ligocki, T.J.; Martin, D.; McCorquodale, P.; Modiano, D.; Schwartz, P.O.; et al. *Chombo Software Package for AMR Applications—Design Document*; Applied Numerical Algorithms Group Computational Research Division Lawrence Berkeley National Laboratory: Berkeley, CA, USA, 2015.
- Rossinelli, D.; Hejazialhosseini, B.; Spampinato, D.G.; Koumoutsakos, P. Multicore/multi-GPU accelerated simulations of multiphase compressible flows using wavelet adapted grids. *SIAM J. Sci. Comput.* **2011**, *33*, 512–540. [\[CrossRef\]](#)
- Kevlahan, N.K.R.; Dubos, T. WAVETRISK-1.0: An adaptive wavelet hydrostatic dynamical core. *Geosci. Model Dev.* **2019**, *12*, 4901–4921. [\[CrossRef\]](#)
- Paolucci, S.; Zikoski, Z.J.; Wirasaet, D. WAMR: An adaptive wavelet method for the simulation of compressible reacting flow. Part I. Accuracy and efficiency of algorithm. *J. Comput. Phys.* **2014**, *272*, 814–841. [\[CrossRef\]](#)
- Paolucci, S.; Zikoski, Z.J.; Grenga, T. WAMR: An adaptive wavelet method for the simulation of compressible reacting flow. Part II. The parallel algorithm. *J. Comput. Phys.* **2014**, *272*, 842–864. [\[CrossRef\]](#)
- Schroeder, W.; Martin, K.; Lorensen, B. *The Visualization Toolkit*, 4th ed.; Kitware: Clifton Park, NY, USA, 2006.
- Ayachit, U. *The ParaView Guide: A Parallel Visualization Application*; Kitware: Vienna, Austria, 2015.
- Childs, H.; Brugger, E.; Whitlock, B.; Meredith, J.; Ahern, S.; Pugmire, D.; Biagas, K.; Miller, M.; Harrison, C.; Weber, G.H.; et al. VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In *High Performance Visualization—Enabling Extreme-Scale Scientific Insight*; CRC Press: Boca Raton, FL, USA, 2012; pp. 357–372. [\[CrossRef\]](#)
- Ligocki, T. Implementing a Visualization Tool for Adaptive Mesh Refinement Data using VTK. In Proceedings of the Visualization Development Environments, Princeton, NJ, USA, 27–28 April 2000.
- Wald, I.; Brownlee, C.; Usher, W.; Knoll, A. CPU Volume Rendering of Adaptive Mesh Refinement Data. In Proceedings of the SIGGRAPH Asia 2017 Symposium on Visualization, Los Angeles, CA, USA, 30 July–3 August 2017; ACM: New York, NY, USA, 2017; pp. 9:1–9:8. [\[CrossRef\]](#)
- Ma, K.L.; Crockett, T. A scalable parallel cell-projection volume rendering algorithm for three-dimensional unstructured data. In Proceedings of the IEEE Symposium on Parallel Rendering (PRS'97), Phoenix, AZ, USA, 20–21 October 1997; pp. 95–104. [\[CrossRef\]](#)
- Kaehler, R.; Hege, H.C. Texture-based volume rendering of adaptive mesh refinement data. *Vis. Comput.* **2002**, *18*, 481–492. [\[CrossRef\]](#)
- Kaehler, R.; Wise, J.; Abel, T.; Hege, H.C. *GPU-Assisted Raycasting for Cosmological Adaptive Mesh Refinement Simulations*; Volume Graphics; Machiraju, R., Moeller, T., Eds.; The Eurographics Association: Munich, Germany, 2006. [\[CrossRef\]](#)
- Kaehler, R.; Abel, T. Single-pass GPU-raycasting for structured adaptive mesh refinement data. In *Proceedings of the SPIE, Burlingame, CA, USA, 3–7 February 2013*; Wong, P.C., Kao, D.L., Hao, M.C., Chen, C., Healey, C.G., Eds.; SPIE: Burlingame, CA, USA, 2013. [\[CrossRef\]](#)
- Leaf, N.; Vishwanath, V.; Insley, J.; Hereld, M.; Papka, M.E.; Ma, K.L. Efficient parallel volume rendering of large-scale adaptive mesh refinement data. In Proceedings of the 2013 IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV), Atlanta, GA, USA, 13–14 October 2013; pp. 35–42. [\[CrossRef\]](#)

25. Weber, G.H.; Childs, H.; Meredith, J.S. Efficient parallel extraction of crack-free isosurfaces from adaptive mesh refinement (AMR) data. In Proceedings of the IEEE Symposium on Large Data Analysis and Visualization (LDAV), Seattle, WA, USA, 14–15 October 2012; pp. 31–38. [[CrossRef](#)]
26. Sweldens, W. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.* **1998**, *29*, 511–546. [[CrossRef](#)]
27. Vasilyev, O.V. Solving multi-dimensional evolution problems with localized structures using second generation wavelets. *Int. J. Comput. Fluid Dyn.* **2003**, *17*, 151–168. [[CrossRef](#)]
28. Vasilyev, O.V.; Kevlahan, N.K.R. An adaptive multilevel wavelet collocation method for elliptic problems. *J. Comput. Phys.* **2005**, *206*, 412–431. [[CrossRef](#)]
29. Muraki, S. Approximation and rendering of volume data using wavelet transforms. In Proceedings of the Visualization '92, Boston, MA, USA, 19–23 October 1992; pp. 21–28. [[CrossRef](#)]
30. Lippert, L.; Gross, M.; Kurmann, C. Compression Domain Volume Rendering for Distributed Environments. *Comput. Graph. Forum* **1997**, *16*, C95–C107. [[CrossRef](#)]
31. Westermann, R. A Multiresolution Framework for Volume Rendering. In Proceedings of the 1994 Symposium on Volume Visualization, Washington, DC, USA, 17–18 October 1994; Association for Computing Machinery: New York, NY, USA, 1994; pp. 51–58.
32. Guthe, S.; Wand, M.; Gonser, J.; Strasser, W. Interactive rendering of large volume data sets. *IEEE Vis.* **2002**, *2002*, 53–60.
33. Welsh, T.; Mueller, K. A Frequency-Sensitive Point Hierarchy for Images and Volumes. *IEEE Vis.* **2003**, *2003*, 425–432. [[CrossRef](#)]
34. Lippert, L.; Gross, M.H. Fast Wavelet Based Volume Rendering by Accumulation of Transparent Texture Maps. In *Computer Graphics Forum*; Blackwell Science Ltd.: Edinburgh, UK, 1995. [[CrossRef](#)]
35. Gross, M.; Lippert, L.; Dittrich, R.; Häring, S. Two methods for wavelet-based volume rendering. *Comput. Graph.* **1997**, *21*, 237–252. [[CrossRef](#)]
36. Bajaj, C.; Ihm, I.; Koo, G.B.; Park, S. *Parallel Ray Casting of Visible Human on Distributed Memory Architectures*; Gröller, E., Löffelmann, H.; Ribarsky, W., Eds.; Springer: Vienna, Austria, 1999; pp. 269–276.
37. Nguyen, K.G.; Saupe, D. Rapid High Quality Compression of Volume Data for Visualization. In *Computer Graphics Forum*; Blackwell Science Ltd.: Edinburgh, UK, 2001; Volume 20.
38. Lippert, L.; Gross, M.H. *Ray-Tracing of Multiresolution B-Spline Volumes*; Technical Report 239; Computer Science Department, ETH: Zürich, Switzerland, 1996.
39. Yu, H.; Chang, E.C.; Huang, Z.; Zheng, Z. Fast Rendering of Foveated Volume in the Wavelet Domain. In Proceedings of the IEEE Visualization 2004, Austin, TX, USA, 10–15 October 2004; 26p. [[CrossRef](#)]
40. Freitag, L.; Loy, R. Adaptive, Multiresolution Visualization of Large Data Sets using a Distributed Memory Octree. In Proceedings of the SC '99: Proceedings of the 1999 ACM/IEEE Conference on Supercomputing, Portland, OR, USA, 13–19 November 1999; p. 60.
41. Kreylos, O.; Weber, G.H.; Wes, E.; John, B.; Shalf, M.; Hamann, B.; Joy, K.I. *Remote Interactive Direct Volume Rendering of AMR Data*; Technical report; Lawrence Berkeley National Laboratory: Berkeley, CA, USA, 2002.
42. Weber, G.H.; Kreylos, O.; Ligocki, T.J.; Shalf, J.; Hagen, H.; Hamann, B.; Joy, K.I.; Ma, K.L. *High-Quality Volume Rendering of Adaptive Mesh Refinement Data*; Vision, Modeling and Visualization; Aka GmbH: Augsburg, Germany, 2001; pp. 121–128.
43. Lacroute, P.; Levoy, M. Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation. In Proceedings of the SIGGRAPH '94, Orlando, FL, USA, 24–29 July 1994; Association for Computing Machinery: New York, NY, USA, 1994; pp. 451–458.
44. Lacroute, P. *The VolPack Volume Rendering Library*; Stanford University: Stanford, CA, USA, 1995.
45. Schroeder, W.; Martin, K.; Lorensen, W. The design and implementation of an object-oriented toolkit for 3D graphics and visualization. In Proceedings of the Seventh Annual IEEE Visualization '96, San Francisco, CA, USA, 27 October–1 November 1996; pp. 93–100. [[CrossRef](#)]
46. Schroeder, W.; Geveci, B.; Malaterre, M. Compatible triangulations of spatial decompositions. *IEEE Vis.* **2004**, *2004*, 211–217. [[CrossRef](#)]
47. *Amira-User's Guide and Reference Manual*; Zuse Institute Berlin (ZIB) and Indeed-Visual Concepts GmbH: Berlin, Germany, 2001.