Practical Reconstruction and Hardware-Accelerated Direct Volume Rendering on Body-Centered Cubic Grids

Oliver Mattausch*

Institute of Computer Graphics and Algorithms
Vienna University of Technology
Vienna / Austria

Abstract

In volume visualization, the Cartesian grid is by far the most popular type of grid because it is convenient to handle. But it requires 29.3% more samples than the Body-Centered Cubic grid. In order to convince people used to Cartesian grids for years of the advantages of Body-Centered Cubic grids, we must prove their usability in many different volume rendering algorithms. Therefore we introduce several practical reconstruction schemes on Body-Centered Cubic grids, which are very general and can be used in a number of applications and tasks.

Together with the development of powerful and flexible consumer graphics hardware, interactive hardware-accelerated volume rendering algorithms gain popularity. Rendering performance becomes a big issue, which can be a strong argument in favour of Body-Centered Cubic grids. We adapted the projected tetrahedra algorithm to Body-Centered Cubic grids, which is one of the most popular volume rendering approaches exploiting hardware-acceleration. At least partly we succeeded in achieving a performance gain on our new grid and further produced some impressive rendering results comparable to the Cartesian grid version.

Keywords: hardware acceleration, alternative grids, ray-casting, cell projection, hexagonal, sphere packing

1 Introduction

The Cartesian (CC) grid is the dominant type of sampling grid in volume visualization. Although we know from signal theory that it is not optimal in terms of storage efficiency [18]. To find the optimal sampling grid, we have to solve the dual problem of packing the replicated spectra in the frequency domain as closely as possible so that they do not overlap. In volume rendering, it is assumed that we are dealing with isotropic, band-limited scalar functions which have spherical spectra. Our problem is equivalent to the famous sphere packing problem [14]. There is no general solution to this problem yet, but fortunately for our purpose some optimal packing schemes among regular grids in 2D and 3D are known.

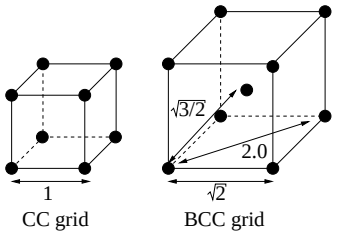

Figure 1: CC and BCC cells in relative proportions (image redrawn from Theußl et al. [17]).

In 3D the Hexagonal Close Packing (HCP) grid and the Face-Centered Cubic (FCC) grid are optimal sphere packings. The dual of the HCP grid in the spatial domain is again a HCP grid, the dual of the FCC grid is the Body-Centered Cubic (BCC) grid. The BCC grid looks like a CC grid with an additional sample point in the middle of each cubic cell (see figure 1). Both schemes require 29.3% less samples than the CC grid to store the same amount of information [18]. The BCC grid is easier to handle than the HCP grid, because it can be described by a sampling matrix as opposed to the HCP grid, and has some convenient properties that can be exploited for volume rendering. The BCC grid can be seen as

- stack of 2D CC grids, where the odd-numbered planes are translated by half a unit in both dimensions with respect to the even-numbered planes.
- two interleaved 3D CC grids, where one grid (denoted as secondary grid) is translated by half a cell spacing in all three axes relatively to the other grid (denoted as primary grid).
- sheared and scaled CC grid.
- tetrahedral mesh, which is uniquely defined by the Delauney complex [2].

Our goal is to show to the volume visualization community that the BCC grid is an alternative to the CC grid in practice. Therefore we must first proof it's usability in different types of volume rendering algorithms. Furthermore,

*matt@cg.tuwien.ac.at

we have to achieve a performance gain over most comparable rendering approaches on the CC grid due to the reduced memory requirements, with equal or only slightly reduced image quality. In chapter 3, we introduce some practical schemes for reconstruction in a BCC grid.

An evolving field of volume visualization deals with exploiting the power of flexible consumer graphics hardware for interactive or near-interactive volume rendering. Hardware acceleration could increase the popularity of traditionally slow direct volume rendering for usage in time-critical applications as well. The potential performance gain of BCC grids is therefore even more desirable in such hardware-based approaches. An important hardware-accelerated method is the projected tetrahedra algorithm. In chapter 4, we present a speed-optimized implementation of the algorithm for the BCC grid.

2 Previous work

Volume rendering can be classified into image-order algorithms like raytracing and raycasting [7], and object-order algorithms like splatting [20], cell projection [13], and texture-based volume rendering [1]. A hybrid method is the performance-optimized shear-warp algorithm introduced by Lacroute et al. [6].

The projected tetrahedra algorithm proposed by Shirley et al. [13] exploits hardware-acceleration for the rendering of any type of tetrahedral grid. Accuracy of the transparency calculation was improved by Stein et al. [15]. Accurate integration of arbitrary transfer functions is possible by using the pre-integration technique introduced by Röttger et al. [11].

Theußl et al. [18] first proposed the usage of BCC grids for direct volume rendering. They implemented Westover style splatting [20] on the BCC grid, which was extended to the 4^{th} dimension for time-varying data by Neophytou and Mueller [9]. The shear-warp algorithm was extended to support BCC rendering by Sweeney et al. [16]. Ibáñez et al. [5] used a generalization of the Bresenham algorithm for raycasting on the BCC grid, but they did not present any details about the used interpolation. Reconstruction schemes for high-quality raycasting on the BCC grid were proposed by Theußl et al. [17]. In chapter 3, we present their methods in detail. Dornhofer modified Fourier Domain Volume Rendering (FDVR) for use on a BCC grid [4]. Iso-surface reconstruction on the tetrahedral mesh defined by a BCC grid was proposed by Chan and Purisma [3]. To cope with the large number of created triangles on such a mesh, Carr et al. [2] investigated and compared the marching cubes variants marching tetrahedra, marching octahedra, and marching hexahedra for BCC grid iso-surface reconstruction.

For a more detailed description of the methods presented in this work, refer to the diploma thesis of Mattausch [8]. In this thesis, the adaption of other hardware-accelerated approaches (e.g., 2D and 3D texture-based volume rendering) to the BCC grid is also described.

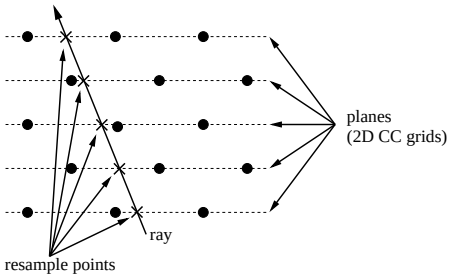

Figure 2: Bilinear interpolation in the 2D CC grid planes, which are most perpendicular to the viewing direction (image taken from Theußl et al. [17]).

3 Reconstruction Schemes

We developed some general strategies for practical reconstruction on the BCC grid. The reconstruction schemes were integrated in a raycasting system for high-quality rendering. However, they can be used in other BCC rendering algorithms which need an interpolation between sample positions. Some of the reconstruction schemes are optimized for speed, others for rendering quality, while all preserve reasonable complexity.

There is one restriction to our methods. The quality of some techniques depends on the current view direction. As a consequence, these methods are only suited for applications with a defined view direction, like volume rendering. They are not suited for view-independent applications, like segmentation.

3.1 Bilinear Interpolation

Bilinear interpolation operates directly on the 2D CC grid planes. On the BCC grid, every second plane is translated by half a unit. If we use bilinear interpolation for raycasting, we must ensure that the entry point of the ray is also on a plane [19]. From the three stacks of resampling planes, we choose the planes most perpendicular to the current view direction (shown in figure 2). On the BCC grid we get a higher sample frequency than on the CC grid, because the planes are closer together by a factor of $\sqrt{2}$. On the other hand, we lose information in the planes, because they consist of half the number of samples than on the CC grid.

To further halve the sample distance, we can apply a simplified trilinear interpolation directly in between two planes [19]. This interpolation is a specialized version of the sheared trilinear interpolation from section 3.4.

3.2 Bilinear plus Spatial Interpolation

Bilinear interpolation can be extended to a real trilinear interpolation on arbitrary resampling positions. This can

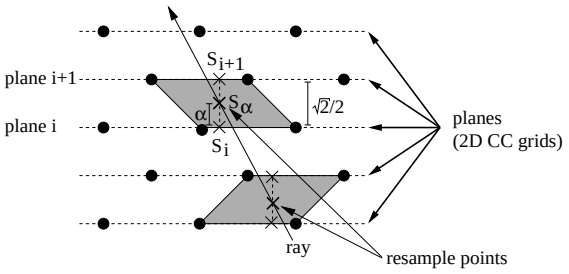

Figure 3: Bilinear plus spatial interpolation shown in 2D. After we bilinearly interpolate the values from the planes (S_i and S_{i+1}), a linear interpolation with α as weight yields the final scalar value S_α.

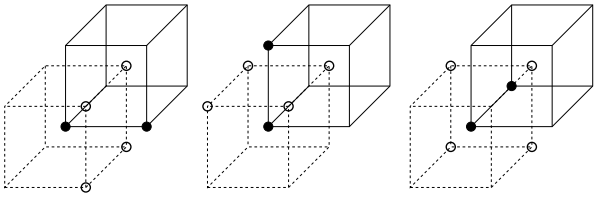

Figure 4: The Delaunay tetrahedralization of the BCC grid. A tetrahedron is determined by two samples from the primary (secondary) grid (black dots) together with two adjacent samples from the secondary (primary) grid (white dots). Image taken from Theußl et al. [17].

be achieved with an additional spatial interpolation of two bilinearly interpolated density values on the adjacent 2D CC grid planes. Therefore we refer to this reconstruction scheme as bilinear plus spatial interpolation. The situation is depicted in figure 3, where the interpolated values from the upper and lower plane are denoted as S_i and S_{i+1}, and α refers to the spatial distance from the planes. We calculate the final density value S_α like the following:

$$S_\alpha = (\frac{\sqrt{2}}{2} - \alpha)S_i + \alpha S_{i+1} \qquad \text{for } 0 < \alpha < \frac{\sqrt{2}}{2} \quad (1)$$

Similar to bilinear interpolation, we choose the stack of CC grid planes that is most perpendicular to the actual view direction. This method results in an interpolation in a sheared cubic cell, where the shear directions of the cells are determined by the resampling location. The grey areas in figure 3 show the cells in 2D.

3.3 Barycentric Interpolation

Barycentric interpolation operates on the tetrahedral mesh that is defined by the BCC grid. The barycentric coordinates are calculated, and used for an interpolation between the tetrahedron vertices. The interpolation between the vertices is piece-wise linear and therefore fast, but of limited quality. Additional computations must be done to find the current tetrahedron that contains the resampling point, and to calculate the barycentric coordinates:

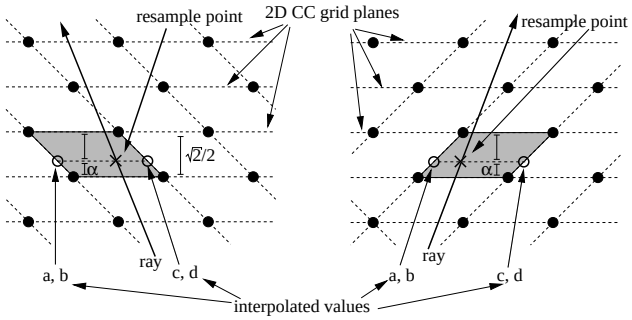

Figure 5: 2D visualization of sheared trilinear interpolation in the BCC grid. The linearly interpolated values a, b, c, d are used for a bilinear interpolation.

1. Find the tetrahedron where the resampling point is located in. First we determine the corresponding octant of the cell in the primary (secondary) grid using three comparisons. We need some more comparisons (x greater or smaller y, x greater or smaller z, and y greater or smaller z) to find the current tetrahedron. The situation is visualized in figure 4.

2. Compute the barycentric coordinates. This is done by transforming the resampling point into a coordinate system, where one vertex of the tetrahedron is in the origin, and the others are in unit distance on the x, y, and z axis. The barycentric coordinates are equivalent to the new location of the resampling point. After translation to the origin, only 12 different types of tetrahedra exist, thus the transformation matrices can be precomputed and stored in a table.

3.4 Sheared Trilinear Interpolation

The BCC grid can be seen as sheared and scaled CC grid, where we operate on sheared cubic cells. We can use a special kind of trilinear interpolation in these cells. which we denote as sheared trilinear interpolation. A 2D visualization is depicted in figure 5. The cells are sheared along two axes that span 2D CC grid planes. In figure 5, these are the planes made up by the axes pointing to the right and (not seen in the 2D visualization) into the spatial dimension. We denote these planes as shear planes. First we apply linear interpolations along the four short edges of a particular sheared cubic cell. Again in figure 5, the resulting interpolated values are referred as a, b, c, and d. The weights used in the linear interpolations are determined by the distances α and $\frac{\sqrt{2}}{2} - \alpha$ from the adjacent 2D CC grid planes. The scalar values a, b, c, and d can then be used for a bilinear interpolation of the final density value. In this approach, we can freely choose the shear planes and the shear directions of the cells. We made following considerations about the proper selection of these parameters:

- As shear planes, we choose the planes which are most perpendicular to the actual view direction. This

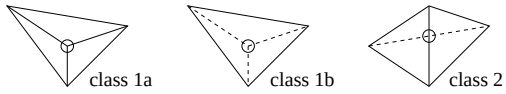

Figure 6: Basic classes for the decomposition of a tetrahedron. The circle refers to the ”thick” vertex.

is closely connected to the bilinear reconstruction scheme from section 3.1. Figure 5 illustrates this approach in 2D.

- Next, we also choose the shear directions depending on the viewing ray direction. The cells are sheared either into the positive or negative directions, in order to assure that the sheared cell borders are as parallel as possible to the ray. This assures that the ray will pass through the sheared cell as similar as possible as it would pass through the original cubic cells. As we can see in the right image of figure 5, the shear directions towards the positive axis would be chosen, if the viewing ray pointed slightly towards the right.

4 Projected Tetrahedra Method

The projected tetrahedra algorithm [13] is a simple and flexible algorithm which uses the properties of a tetrahedral cell. The graphics hardware is exploited to interpolate the scalar function between the vertices. The method consists of the following steps:

1. Decompose the volume into a tetrahedral mesh. Density values are stored at each vertex. The scalar function is assumed to be a linear combination of the vertex values.
2. Depth sort the tetrahedra.
3. Classify tetrahedra and decomposite into triangles according to the projected profile. The two main cases are shown in figure 6.
4. Determine color and opacity values at the triangle vertices using ray integration at the ”thick” vertex.
5. Rasterize the triangles.

Implementation on the BCC grid is straightforward as the tetrahedral mesh is defined by the Delauney tetrahedralization (refer to figure 4). We concentrated on the exploitation of the regular grid structure to speed-optimize the projected tetrahedra method for BCC grids.

4.1 Back-to-Front Traversal

On regular grid structures like the CC and BCC grid, we can do the depth-sort step implicitly with a back-to-front traversal of the sample points. The tetrahedra are created on the fly. For this purpose, we define a cell structure

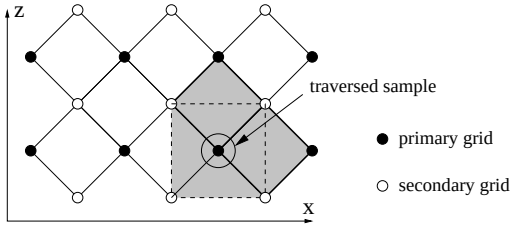

Figure 7: Traversal step of the projected tetrahedra algorithm on the BCC grid (in 2D). In each step three octahedra (shown in grey, the one in the y axis outlined with dotted lines) are created with adjacent samples in the directions of the positive x, y, and z axis, then tetrahedralized.

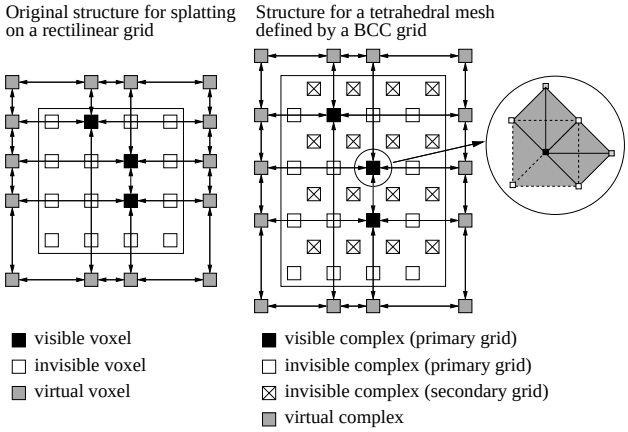

Figure 8: The adjacency structure in 2D. The structure is encapsulated by a box of virtual voxels (traversal complexes) which are only created on demand.

that contains all tetrahedra processed in one traversal step, which we denote as traversal complex. This traversal complex is trivially given by a cube on the CC grid. On the BCC grid, it can be verified that we cover all tetrahedra of the grid by traversing only the primary grid, and tetrahedralizing the three octahedra spanned between the current sample, the adjacent samples of the primary grid in the positive x, y, and z axis (the black dots in figure 4) and four samples from the secondary grid (the white dots in figure 4). A single traversal step is shown in figure 7. For depth-ordering octahedra, back-to-front traversal suffices. Thus we process 12 tetrahedra per traversal step (4 per octahedron). Depth-ordering the tetrahedra inside a traversal complex is trivial. It can be done in a pre-processing step for orthogonal projection, because we have only 12 different types of tetrahedra on the BCC grid.

4.2 Adjacency Structure for Tetrahedral Grids

To further speed up the traversal, we adapted an adjacency data structure proposed by Orchard et al. [10] to store the tetrahedral mesh. This structure was originally used to accelerate splatting in rectilinear grids. It stores only visi-

ble voxels (i.e., the opacity exceeds a certain threshold). Together with a voxel six pointers to the adjacent visible voxels in all axes are stored, allowing to skip transparent voxels completely. A 2D version of the structure is shown in the left image of figure 8. To enable skipping of larger volume regions, a hierarchy of three types of virtual voxels is introduced (denoted as box corner, box edge, and box face voxels [10]), which encapsulate the structure like a box. Box face voxels are stored on each end of visible (i.e., at least one of the voxels is visible) voxel scanlines. Box edge voxels are capping both ends of visible box face scanlines, and visible box edge scanlines are capped by two box corner voxels.

To use the data structure for a tetrahedral mesh, we store traversal complexes instead of voxels. The new structure is depicted in the right image of figure 8. A traversal complexes is marked as visible if at least one of the tetrahedra inside is visible. For each tetrahedron of a visible traversal complex, we must explicitly check for visibility. Fortunately, the visibility information can be efficiently stored together with a traversal complex as a checksum. Each of the tetrahedra is given an unique ID number. The checksum is the sum of the ID numbers of all visible tetrahedra. Hence visibility testing is easily done during traversal by masking this checksum with the unique tetrahedron ID.

4.3 Improving the Rendering Quality

The original projected tetrahedra algorithm assumes a linear color and transparency variation inside a tetrahedron. For linear transfer functions, we can calculate the correct transparency $1 - \exp(-\tau l)$ by applying 2D exponential transparency textures [15]. The extinction coefficient τ and the segment length l are taken as texture coordinates.

For abitrary transfer functions, the pre-integration technique [11] allows accurate renderings without sacrificing hardware acceleration. A 3D texture is used to store pre-integrated ray segments, taking the entry point s_f, the exit point s_b and the segment length l of a ray as parameters. Pre-integration can be seen as an integration of the transfer function separate from the integration of the scalar field.

4.4 Shading Issues

Directional shading is a very important issue in volume rendering, because it enhances the spatial perception and provides useful cues about the shape of the rendered object. For introducing directional shading into the original projected tetrahedra algorithm, we store the normals with a vertex, and apply standard OpenGL Gouraud shading.

It is difficult to use directional shading in combination with pre-integration, because only three parameters can be used as indices into a 3D texture. We introduce two approaches to combine pre-integration and shading, which we denote as pre-integration and Gouraud shading, and pre-integration and Phong shading. However, both approaches do not insert shading information into the pre-integration process.

Dataset	CC Dimension	BCC Dimension
Cube	$40 \times 40 \times 40$	$28 \times 28 \times 56$
Fuel	$64 \times 64 \times 64$	$45 \times 45 \times 90$
Hipiph	$64 \times 64 \times 64$	$45 \times 45 \times 90$
Device	$128 \times 128 \times 64$	$91 \times 91 \times 91$

Table 1: The datasets used in our experiments.

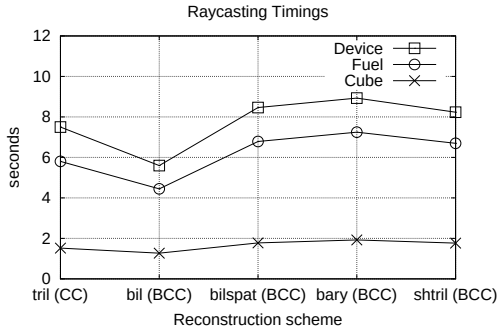

Figure 9: Raycasting timings for different interpolators. "tril" refers to trilinear, "bil" to bilinear, "bilspat" to bilinear plus spatial, and "shtril" to sheared trilinear interpolation.

To combine pre-integration with Gouraud shading, we set the vertex color to white and apply the standard OpenGL Gouraud shading. The resulting grey value represents the Gouraud shaded light intensity in a pixel and is then modulated with the 3D pre-integration texture. To preserve the highlights, we must assure that the specular output is added after the modulation.

To use pre-integration and Phong shading, we store the normals in the color portion, in order to get the interpolated gradients in the pixel shaders. The gradient and the output texel of the pre-integration texture can be employed for the calculation of the Phong shading equation. If there is no square root operation available in the pixel shaders, we can either use a (rather slow) cube map for normalization, or use an approximation formula that is reasonable accurate under the given conditions [12].

5 Results

The datasets used in our experiments are listed in table 1. The Fuel and the Hipiph datasets have been resampled from the CC to the BCC grid format, which is a source for resampling errors. The Device dataset is scanned and sampled on both CC and BCC grid separately. We tested the reconstruction schemes in a raycasting system.

Figure 11 shows renderings of the Fuel dataset on the BCC grid. The bilinear interpolation (left image) reveals slicing artifacts caused by undersampling. The artifacts disappear when using the bilinear plus spatial interpolation

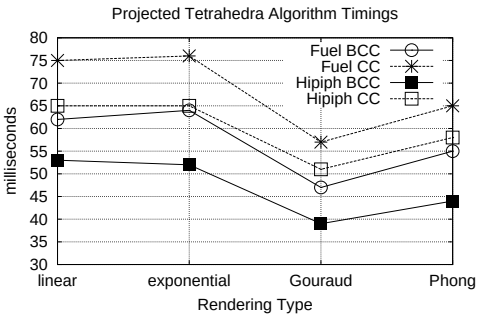

Figure 10: Timings for the projected tetrahedra algorithm. ”Linear” refers to the linear transparency variation in the original projected tetrahedra method. ”exponential” refers to the exponential transparency textures, ”Phong” and ”Gouraud” to the shading techniques for pre-integration.

(right image).

In figure 12, comparisons of different reconstruction schemes on the Device dataset are depicted. Trilinear interpolation on the CC grid (the images in the leftmost column) produces the most detailed renderings. Barycentric interpolation (the images in the middle column) suffers from artifacts clearly visible in the magnified region. Using sheared trilinear interpolation (the images in the rightmost column), no artifacts are visible, but some details are also smoothed out. We can see some noise in the topmost CC grid image, which is missing in the BCC grid renderings. This difference originates from the sampling process.

Rendering results of the projected tetrahedra algorithm on the Fuel dataset are shown in figure 13. The CC grid images are in the top row, the BCC grid images in the bottom row. No major difference can be noticed between the CC grid and the BCC grid renderings of the Fuel dataset. A poor rendering quality is achieved with the exponential transparency textures (the images in the leftmost column). By employing pre-integration (the images in the middle and rightmost column), the rendering quality improves significantly. Together with the pre-integration method, we used our Gouraud shading technique (the images in the middle column), and our Phong shading technique (the images in the rightmost column). The Gouraud shading looks slightly duller than the Phong shading approach, especially in the highlight regions.

In figure 9, timings of our reconstruction schemes are shown. Clearly, bilinear interpolation is the fastest scheme. Next comes trilinear interpolation on the CC grid. Sheared trilinear and bilinear plus spatial interpolation have similar rendering times. Both are slightly slower than trilinear interpolation on the CC grid. Because of the required additional computation steps (refer to section 3.3), barycentric interpolation performs worse than the higher quality methods on the BCC grid.

Figure 10 shows timings of the projected tetrahedra algorithm, equally optimized on both the BCC grid and the

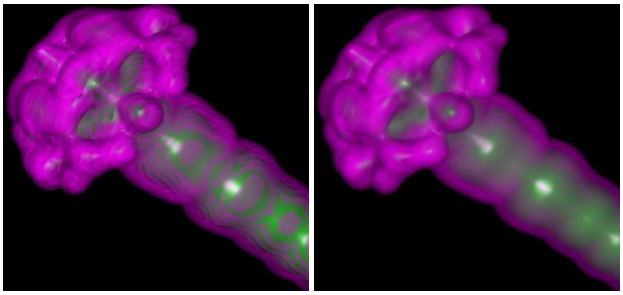

Figure 11: Raycasting on the Fuel dataset for the BCC grid. Bilinear interpolation (left), and bilinear plus spatial interpolation (right).

CC grid. For all tested datasets, the algorithm performs better on the BCC grid volume. Despite using traditionally slow 3D textures, pre-integration is faster than both the original projected tetrahedra algorithm, and the exponential transparency textures. This is because the bottleneck of the algorithm are the per-tetrahedron software computations, which are slightly less complex for the pre-integration approach.

6 Conclusions and Future Work

We presented several practical reconstruction strategies on the BCC grid and tested them in a raycasting system. Among them are fast methods with reasonable quality (e.g., bilinear interpolation), and higher quality methods comparable to the CC grid trilinear interpolation (e.g., bilinear plus spatial, and sheared trilinear interpolation). We achieved the best results when using the sheared trilinear interpolation, or the bilinear plus spatial interpolation. Both schemes are slightly more complex than trilinear interpolation on the CC grid.

Furthermore, we presented a speed-optimized version of an important hardware-accelerated algorithm (i.e., the projected tetrahedra method) for the BCC grid, by exploiting the regular grid structure and orthogonal projection. We adapted a 3D adjacency data structure originally used for splatting to store a tetrahedral mesh defined by a BCC grid (or alternatively derived from the decomposition of a CC grid). This structure allows fast traversal by skipping large regions of transparent tetrahedra.

The tetrahedral mesh defined by a BCC grid consists of a smaller number of tetrahedra than any decomposition of an equivalent CC grid. The smaller number of tetrahedra contributes to a noticeable better performance of the projected tetrahedra algorithm on the BCC grid. This is caused by the fact that the algorithm is a pure object-order technique, hence the number of primitives is responsible for the performance. To produce accurate renderings with projected tetrahedra methods, the pre-integration technique must be used.

Using our approaches, we achieved some impressive

rendering results on the BCC grid. However, we observed a reduced rendering quality on some of our datasets like the Device dataset, regardless of the used method. The renderings are either not equally sharp and detailed, or otherwise have a rather rough appearance.

To find the reason for this quality difference, we must intensively investigate the frequency domain. We developed reconstruction schemes that are very different from the ideal reconstruction filter (i.e., the sinc filter). This is a possible explanation for the BCC grid theory to fail in practice. As a consequence, we should investigate higher-order reconstruction filters for rendering on the BCC grid.

7 Acknowledgments

Thanks to Thomas Theußl, Eduard Gröller, and Markus Hadwiger for their support on this work, and to Anna Vilanova for providing us useful datasets. For further information see http://www.cg.tuwien.ac.at/∼matt/.

References

[1] B. Cabral, N. Cam, and J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings of the 1994 Symposium on Volume Visualization*, pages 91–98. ACM SIGGRAPH, October 1994.

[2] H. Carr, T. Theußl, and T. Möller. Isosurfaces on optimal regular samples. In *Proceedings of the 2003 Joint Eurographics - IEEE TCVG Symposium on Visualization*, pages 39–48, 2003.

[3] S. Chan and E. Purisima. A new tetrahedral tesselation scheme for isosurface generation. In *Computers & Graphics*, volume 22(1), pages 83–90, February 1998.

[4] A. Dornhofer. A discrete fourier transform pair for arbitrary sampling geometries with applications to frequency domain volume rendering on the body-centered cubic lattice. Master's thesis, Vienna University of Technology, 2003.

[5] L. Ibáñez, C.Hamitouche, and C.Roux. Ray casting in the BCC grid applied to 3D medical image visualization. *Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 20(2):548–551, 1998.

[6] P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proceedings of SIGGRAPH '94*, pages 451–458. ACM SIGGRAPH, July 1994.

[7] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, May 1988.

[8] O. Mattausch. Practical reconstruction schemes and hardware-accelerated direct volume rendering on body-centered cubic grids. Master's thesis, Vienna University of Technology, 2004.

[9] N. Neophytou and K. Mueller. Space-time points: 4d splatting on efficient grids. In *Proceedings of the 2002 IEEE Symposium on Volume Visualization*, pages 97–106. IEEE Press, 2002.

[10] J. Orchard and T. Möller. Accelerated splatting using a 3d adjacency data structure. In *GI 2001*, pages 191–200, June 2001.

[11] S. Röttger, M. Kraus, and T. Ertl. Hardware-accelerated volume and isosurface rendering based on cell-projection. In *Proceedings of the IEEE Visualization '00 (Vis '00)*, pages 109–116. IEEE Computer Society Press, 2000.

[12] G. Schröcker. Hardware accelerated per-pixel shading. In *Proceedings of the CESCG*, pages 233 – 246, 2002.

[13] P. Shirley and A. Tuchman. A polygonal approximation to direct scalar volume rendering. In *Computer Graphics (San Diego Workshop on Volume Visualization)*, pages 63–70, November 1990.

[14] N. Sloane. The sphere packing problem. In *ICM: Proceedings of the International Congress of Mathematicians*, pages 387–396, 1998.

[15] C. Stein, B. Becker, and N. Max. Sorting and hardware assisted rendering for volume visualization. In *Proceedings of the 1994 Symposium on Volume Visualization*, pages 83–89. ACM Press, 1994.

[16] J. Sweeney and K. Mueller. Shear-warp deluxe: The shear-warp algorithm revisited. In *Proceedings of the 2002 Joint Eurographics - IEEE TCVG Symposium on Visualization*, pages 95–104, Barcelona, Spain, May 2002.

[17] T. Theußl, O. Mattausch, T. Möller, and Meister E. Gröller. Reconstruction schemes for high quality raycasting of the body-centered cubic grid. Technical Report TR-186-2-02-11, Vienna University of Technology, Institute for Computer Graphics and Algorithms, December 2002.

[18] T. Theußl, T. Möller, and Meister E. Gröller. Optimal regular volume sampling. In *Proceedings of the IEEE Visualization 2001*, pages 91–98, 2001.

[19] M. Wan, A. Kaufman, and S. Bryson. Optimized interpolation for volume ray casting. *Journal of Graphics Tools: JGT*, 4(1):11–24, 1999.

[20] L. Westover. Footprint evaluation for volume rendering. In *Proceedings of SIGGRAPH '90*, pages 367–376. ACM SIGGRAPH, August 1990.

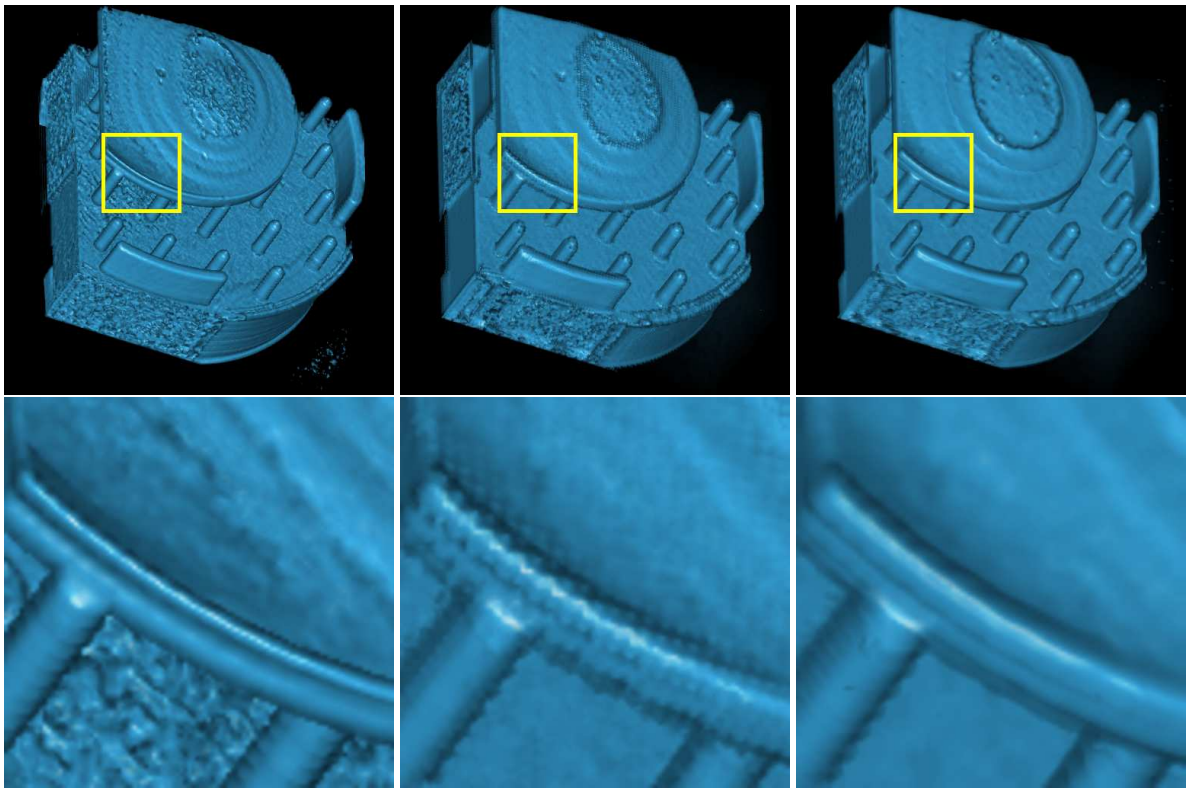

Figure 12: Raycasting on the Device dataset using different interpolators. A zoomed region is shown below each image. From left to right column: trilinear on the CC grid, barycentric on the BCC grid, and sheared trilinear on the BCC grid.

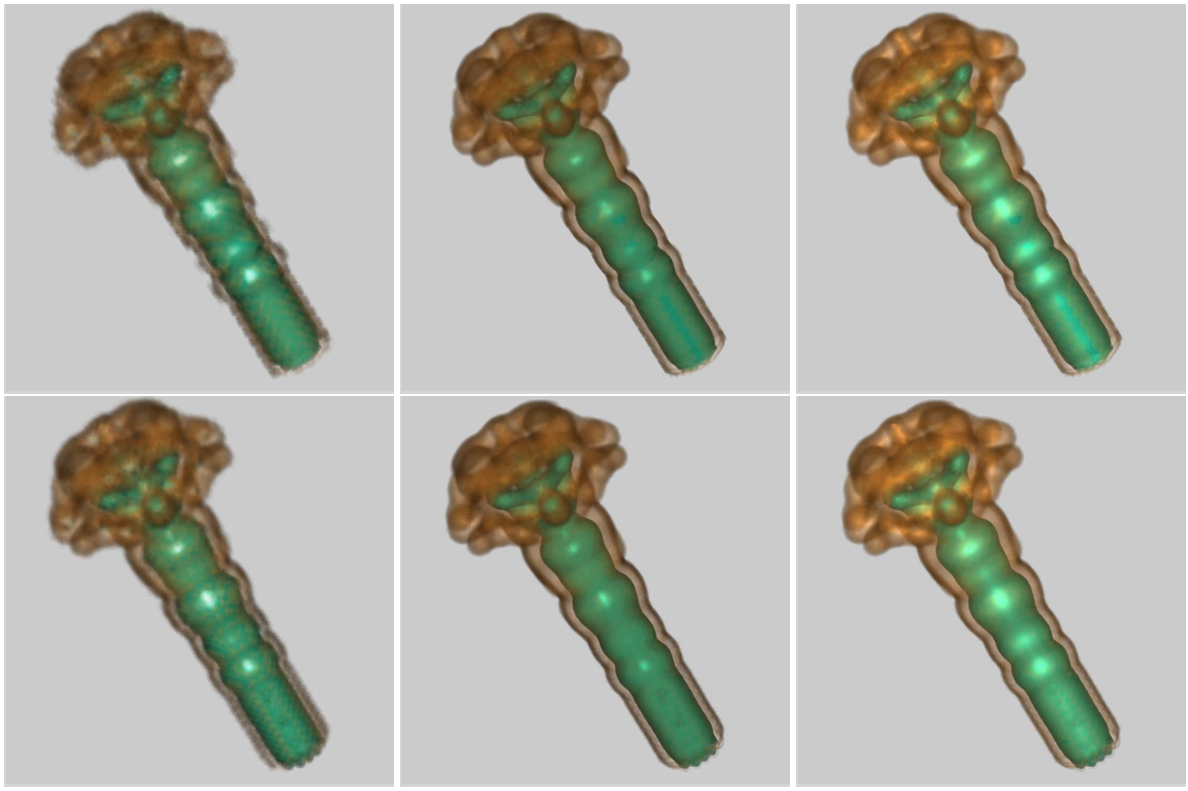

Figure 13: The projected tetrahedra algorithm on the Fuel dataset. The CC grid is shown in the top row, the BCC grid in the bottom row. From left to right column: exponential transparency textures, pre-integration and Gouraud shading, and pre-integration and Phong shading.