Reconstruction Schemes for High Quality Raycasting of the Body-Centered Cubic Grid

Thomas Theußl*, Oliver Mattausch*, Torsten Möller†, and Meister Eduard Gröller*
*Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria
†Graphics, Usability, and Visualization (GrUVi) Lab, Simon Fraser University, Canada

Abstract

The body-centered cubic (BCC) grid has received attention in the volume visualization community recently due to its ability to represent the same data with almost 30% fewer samples as compared to the Cartesian cubic (CC) grid. In this paper we present several resampling strategies for raycasting BCC grids. These strategies range from 2D interpolation in planes to piece-wise linear (barycentric) interpolation in a tetrahedral decomposition of the grid to trilinear and sheared trilinear interpolation. We compare them to raycasting with comparable resampling techniques in the commonly used CC grid in terms of computational complexity and visual quality.

1 Introduction

The main goal of volume rendering is to produce meaningful images of volumetric data. The data usually is sampled into a discrete dataset. This sampling process replicates the frequency response of the original data. If we assume the original data to be isotropic and band-limited, this frequency response is a sphere. In order to avoid aliasing we have to choose a sampling frequency such that the replicas in the frequency domain do not overlap. On the other hand, if we want to use as few samples as possible to describe the data, we have to pack the replicas as closely as possible. In other words, we are facing a sphere packing problem in 3D.

In 1611 Kepler wrote a booklet [11] where he constructed the face-centered cubic (FCC) packing and asserted that no packing of spheres with the same radius in 3D has density greater than the FCC packing. This is known as the Kepler Conjecture. Hilbert formulated it in 1900 as his famous Problem 18 [6]. It is said that Gauß first proved that no regular lattice (i.e., lattices that can be described by three base vectors) is better in this respect [5]. After almost 400 years Hales now claims to have proved it generally [5]. Since we know now that we have to use the FCC grid in the frequency domain, we need the corresponding (dual) grid in the spatial domain for the actual data. This is the BCC grid (see Figure 1) which requires 29.3% fewer samples than the CC grid to represent the same data.
Raycasting [13] is usually considered a rather slow but high quality volume rendering algorithm [16]. With the BCC grid we can reduce the amount of storage needed. However, raycasting is an image order algorithm and we describe exactly the same region of space with the BCC grid instead of the CC grid, only in a different manner. We would therefore expect approximately comparable results on the two grids in terms of execution time.

Besides being the dual of the optimal packing grid in 3D the BCC grid has some more interesting properties, which we will exploit in this paper:

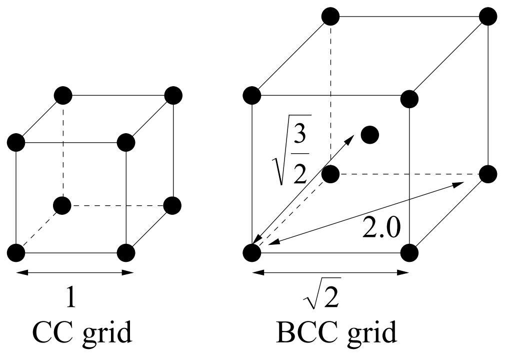

Figure 1: One cell of the CC and the BCC grid in relative proportions. The BCC grid has one additional sample point in the middle of the cell. It requires 29.3% fewer samples to represent the same data.

- It consists of a stack of 2D CC grids where the odd-numbered planes are moved half a unit with respect to the even-numbered planes.
- Similarly, it can also be viewed as two inter-penetrating CC grids (usually called primary and secondary grid), one moved half a unit with respect to the other.
- The Delaunay tetrahedralization is simple and uniquely defined.
- It also results from shearing a CC grid.
- It assumes the frequency response of the originally sampled function to be spherical, consequently, the ideal resampling filter is spherically symmetric.

After we discuss related work in Section 2, we present our approaches to raycast BCC grids in Section 3. We present our results and compare them to each other and to results from the CC grid in Section 4. We then conclude in Section 5 and present ideas for future work in Section 6.

2 Previous Work

Chan and Purisma used the BCC grid as a basis for a tessellation scheme for marching tetrahedra [3], subsequently also used by Treece et al. [23]. Carr et al. [2] reduced the number of triangles generated by adapting the marching tetrahedra algorithm to marching octahedra and marching hexahedra algorithms.

Ibáñez et al. proposed to use the BCC grid because of its optimal topological and spectral properties. They implemented isosurfacing on the BCC and the FCC grid [7] which operates directly on the Voronoi regions of the grids.

*`{theussl|matt|meister}@cg.tuwien.ac.at`, `http://www.cg.tuwien.ac.at/home/`
†`torsten@cs.sfu.ca`, `http://gruvi.cs.sfu.ca/`

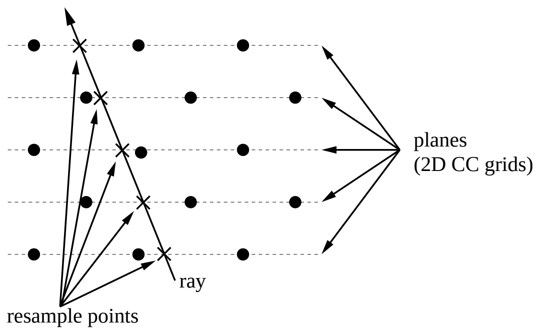

Figure 2: Bilinear interpolation in the planes most perpendicular to the viewing direction. The planes are comprised of 2D CC grids.

Ibáñez et al. also came up with a raycasting algorithm [8, 9]. They adapted the Bresenham line drawing algorithm to the BCC grid (and eventually to arbitrary grids in arbitrary dimensions [10]). This is equivalent to nearest neighbor interpolation. They also discuss barycentric interpolation within tetrahedra, but do not give details about the chosen subdivision or how they identify the tetrahedron the point lies in.

Recently, the BCC grid was employed in volume rendering. Theußl et al. used it to speed up splatting [22]. Neophytou and Mueller adapted this approach to time-varying data [19]. Sweeney and Mueller adapted the shear warp algorithm to the BCC grid [21].

3 Raycasting on the BCC grid

Raycasting usually requires interpolation at arbitrary positions, i.e., a black box that accepts a position and returns an interpolated value. With such an interpolator the sampled dataset can be regarded as being actually continuous. Before we present our approaches to implement such a black box for the body centered cubic grid, we investigate a raycasting scheme using only bilinear interpolation.

3.1 Bilinear Interpolation

Since the BCC grid is a stack of 2D CC grids (i.e., planes) shifted with respect to each other, we can choose the one of three possible stacks which is most perpendicular to the viewing direction and use bilinear interpolation within these planes (see Figure 2). The CC grid is also composed of 2D CC grids which is often exploited in volume visualization, most notably (and efficiently) by the shear warp algorithm [12]. This algorithm can therefore straightforwardly be adapted to the BCC grid [21].

Wan et al. [24] suggest to exploit this to speed up actual raycasting. They do not achieve results comparable in speed to the shear warp algorithm. However, they achieve better quality with the additional possibility of adaptive depth sampling in case two consecutive sample points differ more than a certain threshold. This approach is also straightforwardly adapted to the BCC grid. However, since the step size along the ray is given by the grid and viewing direction, a comparison is only possible indirectly.

Assume we have a dataset sampled on a CC grid with dimensions N^3. Then the corresponding BCC dataset has dimensions [22] $\frac{\sqrt{2}N}{2} \times \frac{\sqrt{2}N}{2} \times \sqrt{2}N$ (w.l.o.g. we assume the third dimension to be most perpendicular to the viewing direction). Let s denote the step size along the ray which is determined by the distance between the slices (Figure 2). Then we will have N sample points with $1 \leq s \leq \sqrt{3}$ in the CC grid and $\sqrt{2}N$ sample points with

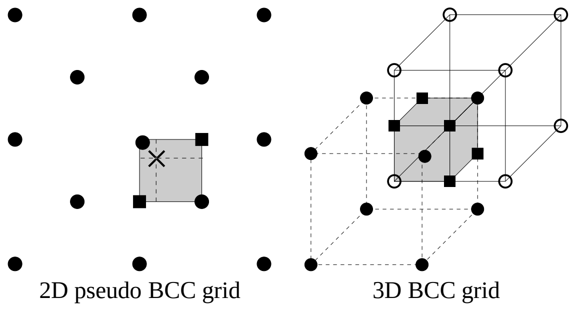

Figure 3: Bilinear interpolation in the 2D pseudo BCC grid on the left and trilinear interpolation in the BCC grid on the right. To interpolate at a position in the shaded box, the missing corners of the box (square dots) are linearly interpolated. Contrary to the 2D pseudo BCC grid this interpolation is uniquely defined in the 3D BCC grid.

$\frac{\sqrt{2}}{2} \leq s \leq \sqrt{\frac{3}{2}}$ in the BCC grid requiring bilinear interpolation. It is usually assumed that the sampling step size should be less than one [24]. So this approach is not appropriate for the CC grid. In the case of the BCC grid, we have certain viewing directions where the sampling step size is less than one and therefore in this cases this approach is sufficient.

In order to have a step size smaller than one, Wan et al. [24] propose to resample intermediate planes on the fly. This resampling halfway in between two planes is also slightly less complex than trilinear interpolation. If we use such intermediate planes we have to resample $N-1$ points with the more complex interpolation in the CC grid and $\sqrt{2}N-1$ points in the BCC grid effectively halving the sampling step size. If two consecutive sample points still differ more than a certain threshold we can adaptively sample in between.

3.2 Trilinear Interpolation

In order to have a black box trilinear interpolation operator similar to the CC grid, we fit the largest possible cube in the BCC grid containing the resampling point which does not contain any other sample points. Alternatively, this can be viewed as taking the intersection of the cubes in the primary and secondary grid containing the resampling point. Figure3 illustrates this for a 2D pseudo BCC grid (where the concept can be seen more clearly, however, this grid is different from the optimal, the hexagonal, grid in 2D and is only shown here and in further examples for demonstration purposes) on the left and the actual 3D BCC grid on the right. For the sake of clarity the actual resampling point has been omitted in the 3D BCC grid.

Two points of this fitted cube are given by the BCC grid (lying opposite of one diagonal of the cube), the other six are linearly interpolated along the edges of the cube. Contrary to the 2D pseudo BCC grid, Figure 3 illustrates that this procedure is uniquely defined in the actual 3D BCC grid. However, since the determination of the cube the point is located in is slightly more complex than in the CC grid and we have to interpolate the missing points, this approach requires slightly more operations than trilinear interpolation in the CC grid. Nevertheless, it is quite easily verified that this scheme results in a continuous, piece-wise cubic interpolator (in accordance with the trilinear interpolator in the CC grid).

Note, that it would also be possible to interpolate the missing points bilinearly on the faces of the cubes. There are essentially two possibilities to do this. We could either replace the linear interpolation by a more costly bilinear interpolation on the fly. Alternatively, we could precompute the missing points and store them in

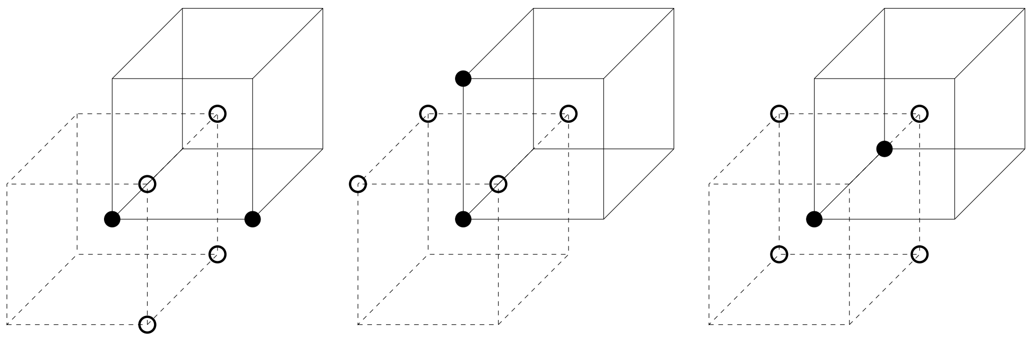
Figure 4: The Delaunay tetrahedralization of the BCC grid. Two adjacent points (white dots) together with the two points of the spine (black dots) in *x*, *y*, and *z* direction (from left to right) make up a tetrahedron. Consequently, there are twelve sets of differently oriented tetrahedra in the Delaunay tetrahedralization of the BCC grid.

an even larger CC grid (and thereby calling in question the use of the BCC grid in the first place). Consequently, we prefer the linear interpolation along the edges of the cubes.

3.3 Barycentric Interpolation

To speed up computations, we investigate barycentric interpolation. This results in a piece-wise linear function along the ray, as opposed to the piece-wise cubic function along the ray with trilinear interpolation. We would therefore also expect a loss in quality.

In order to use barycentric interpolation we perform the following steps:

1. Define a tetrahedralization (mesh) of the BCC grid. An obvious way is to use the Delaunay tetrahedralization. Carr et al. [2] pointed out that this mesh is simple and uniquely defined. Figure 4 visualizes this mesh which consists of twelve sets of differently oriented tetrahedra.

2. Find the tetrahedron the point is located in. We first find the corresponding octant of the cube in the primary (or secondary) grid (the shaded box in Figure 3 on the right) This takes three comparisons. We then need three more comparisons (*x* greater or smaller *y*, *x* greater or smaller *z*, and *y* greater or smaller *z*) to locate the actual tetrahedron.

3. Finally, compute the barycentric coordinates of the resampling point and calculate its value. This is easily done by transforming the tetrahedron into another one with one point in the origin and the other points in unit distance in *x*, *y*, and *z* direction. Since there are twelve different tetrahedra (after translating them to the origin), the twelve different transformation matrices needed can be precomputed.

3.4 Sheared Trilinear Interpolation

Sheared trilinear interpolation takes into account that BCC grids can be seen as a sheared CC grid [4, 22]. Thus every grid cell can be seen as a sheared CC grid cell, i.e., a cube. Standard trilinear interpolation can be done in such cells by first linear interpolating the values on the sheared cell borders, then doing a bilinear interpolation in the resulting square.

However, compared to standard trilinear interpolation additional parameters must be considered, since shear plane and orientation are are not uniquely defined. We apply the following approach to determine shear plane and shear orientation:

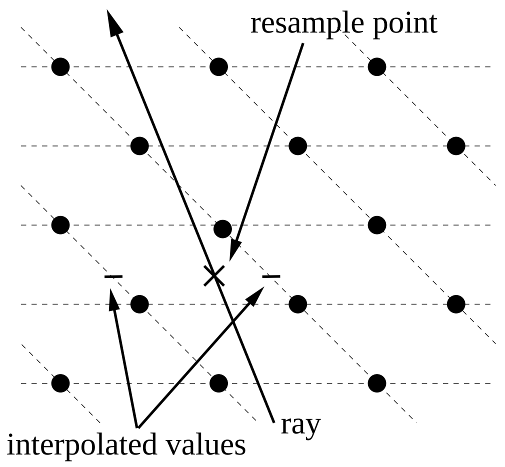

Figure 5: Interpolation in the pseudo 2D BCC grid interpreted as sheared CC grid. The shear axis and orientation are chosen so that they are as closely as possible aligned to the viewing ray direction

- First, we choose that plane as shear plane which is most perpendicular to the viewing ray direction. This is closely connected to the bilinear interpolation approach presented in Section 3.1. The shearing is done in the two axes that span this plane. Figure 5 illustrates this approach in the pseudo 2D BCC grid. If the ray was directed more horizontally, the *y* plane would be chosen as shear plane.

- Next, we also choose the shear orientation depending on the viewing ray direction. The cells are sheared either into positive or negative directions so 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. Again in Figure 5, if the viewing ray pointed slightly towards the right, the shear orientation to the positive *x* direction would be chosen.

The difference to the bilinear interpolation approach is, that we can now freely choose the sampling step. If we choose the same sampling step as in the bilinear interpolation approach, the result will be exactly the same (if the first sampling point is in the same location).

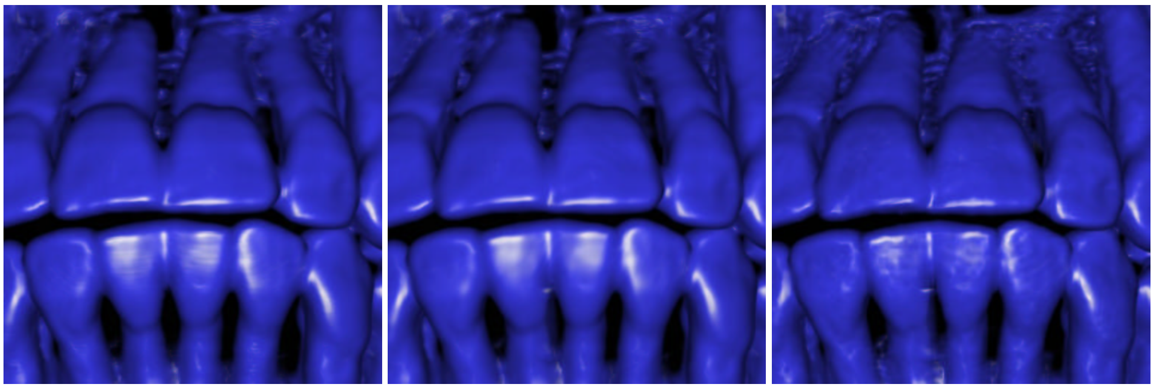

Figure 6: Bilinear (left) and sheared trilinear (middle) on the BCC and trilinear interpolation on the CC (right) grids for the skull dataset

Dataset	CC Dimension	BCC Dimension
Fuel	64 × 64 × 64	45 × 45 × 90
Hipiph	64 × 64 × 64	45 × 45 × 90
M.-Lobb	70 × 70 × 70	49 × 49 × 98
Tooth	256 × 256 × 161	181 × 181 × 227
Skull	256 × 256 × 256	181 × 181 × 362

Table 1: Description of datasets used in our experiments.

4 Comparison

We rendered several datasets with a raycaster represented both on a CC grid and on a BCC grid. Table 1 gives some statistics about all the datasets used in our experiments.

On the CC grid, we used trilinear and bilinear interpolation. On the BCC grid we used the schemes presented in Section 3: bilinear interpolation, trilinear interpolation, barycentric interpolation, and sheared trilinear interpolation. We used the same transfer functions and viewing transformations for all corresponding images. The raycaster does early ray termination and although it supports both pre and post shading, we use post shading for all images.

Figure 6 shows a closeup of the tooth region from the skull dataset and compares bilinear and sheared trilinear interpolation on the BCC grid with trilinear interpolation on the CC grid. The images from the BCC grid appear slightly smoother, visible, for example, at the highlights. This is most likely due to the fact that the BCC dataset was resampled from the CC dataset. The image generated by bilinear interpolation exhibits horizontal artifacts stemming from the plane-wise interpolation. These are completely removed by the sheared trilinear interpolation approach.

Figure 7 shows images from the fuel dataset rendered with all possible interpolators on the BCC grid and bilinear and trilinear interpolation on the CC grid. The images from the bilinear interpolator (middle row) again exhibit artifacts, showing the plane structure of this interpolator, on both grids. The barycentric interpolator on the BCC grid (top left) exhibits artifacts stemming from the piecewise linear nature of this interpolator. The images from the sheared trilinear interpolator on the BCC grid (top right) and the trilinear interpolator on the CC grid (bottom right) show almost no difference.

Figure 8 shows the importance of properly choosing the shear axis for the sheared trilinear interpolator. In the left image the shear axis and orientation were chosen as described in Section 3.4. If the shear axis is chosen differently, as in the middle and right image, artifacts are clearly visible. In these images, to fit the cubic frequency response of the Marschner-Lobb dataset (usually 40 × 40 × 40) into a sphere, we increased the sampling rate to 70 × 70 × 70. Neophytou and Mueller [19] first pointed out that the Marschner-lobb dataset has a clearly non-spherical frequency response. They applied a spherical low-pass filter to compensate for this.

Figures 9 and 10 (top row) show some renderings of the tooth region of the skull dataset and of the tooth dataset, respectively with various interpolators on the BCC and the CC grid. For these rather high resolution datasets, differences between the interpolation schemes and grid types are hardly visible. Figure 10 (bottom row) again compares the interpolation schemes proposed in Section 3 with the hipiph dataset. Although a rather low resolution dataset it is also very smooth, so differences between the schemes are hardly visible.

Tables 2, 3, and 4 show some timing results for the fuel, Marschner-Lobb, and hipiph dataset, respectively. The tables are grouped by sample rate. Within this groups the entries are ordered by rendering time. The image size was 512×512 in all cases. From the timetables we can say that bilinear interpolation is the fastest method. Unfortunately, it cannot be directly compared with bilinear interpolation on CC grids because of the different sample rate due to the cell size difference. Therefore, the groups in the tables either compare bilinear to the other interpolation schemes in the same grid. Otherwise, comparable interpolation schemes in the two different grids are grouped together.

After bilinear interpolation, sheared trilinear and barycentric interpolation are fastest on BCC grids. Sheared trilinear interpolation is slightly faster for the fuel and hipiph datasets, whereas barycentric interpolation is slightly faster for the Marschner-Lobb dataset. Both are also slightly faster than trilinear interpolation on CC grids for the Marschner-Lobb dataset, but slightly slower for the fuel and hipiph datasets.

An interesting result is that, although of lower quality than sheared trilinear, barycentric interpolation is approximately as fast as sheared trilinear interpolation. This is because of the costly inside tetrahedron test and barycentric coordinate calculation. The latter basically involves one translation and a vector-matrix multiplication. Trilinear interpolation on BCC grids is quite slow because of the additional interpolation of the fitted cell points.

5 Conclusions

We presented and compared four different reconstruction schemes for producing high quality raycasting images from scalar data given on BCC grids. Bilinear interpolation basically chooses a step size in such a way that sampling takes place only on cell borders. Trilinear interpolation fits a cube in the BCC grid in order to employ usual trilinear interpolation as on the CC grid. Barycentric interpolation locates the tetrahedron of the Delaunay tetrahedralization of

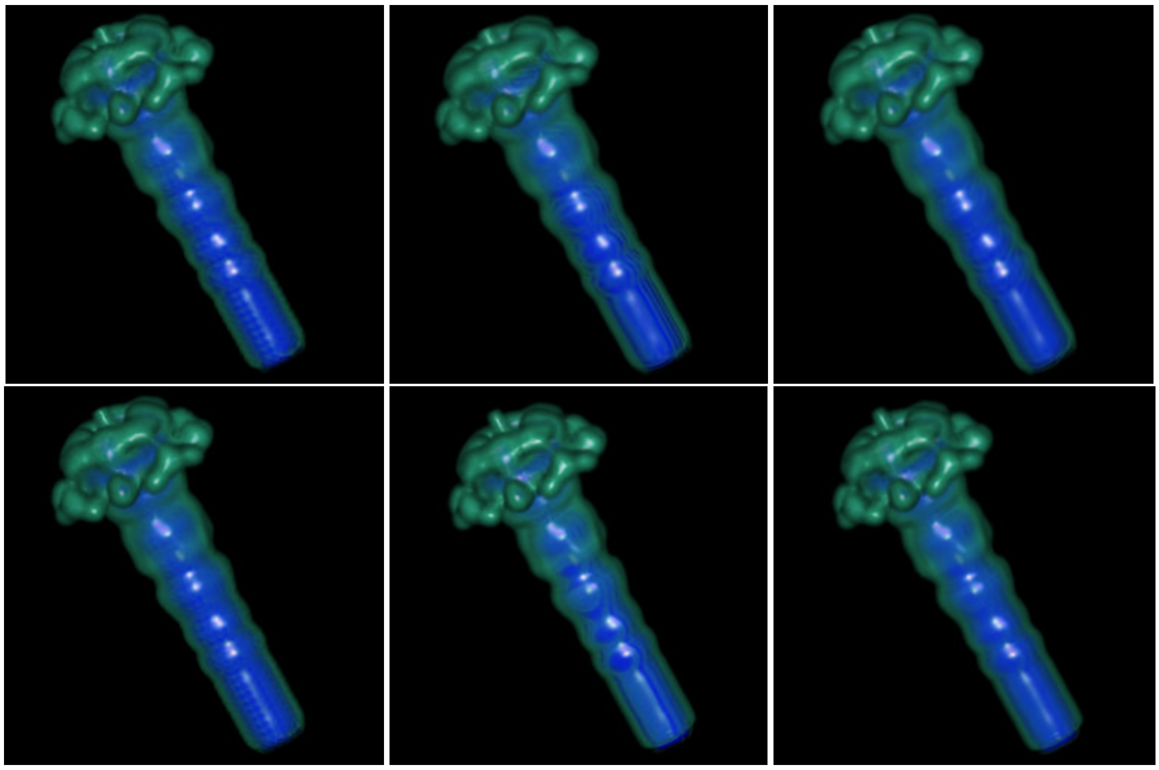

Figure 7: Barycentric, bilinear, sheared trilinear interpolation (top, from left to right), and trilinear on the BCC grid (bottom left) vs. bilinear and trilinear interpolation on the CC grid (bottom, middle and right images)

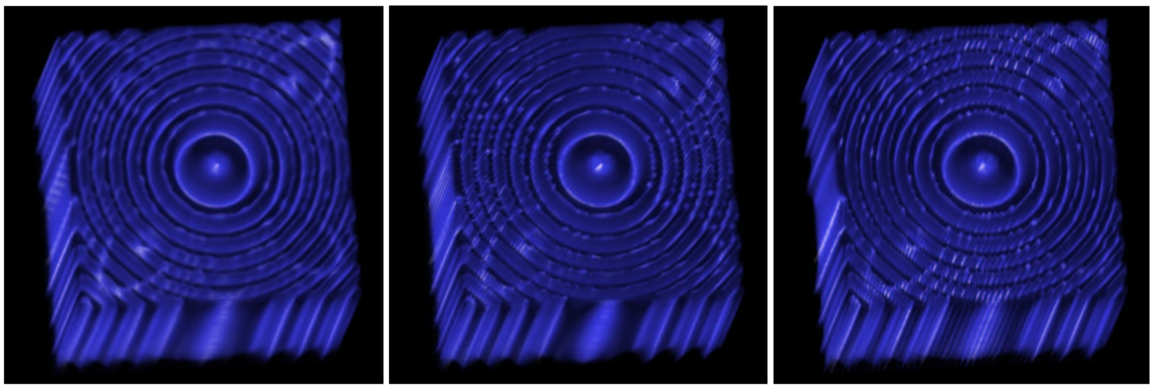

Figure 8: If the shear axis for the sheared trilinear interpolator is chosen dependent on the viewing direction (left image), artifacts are removed which can occur otherwise (middle and right images).

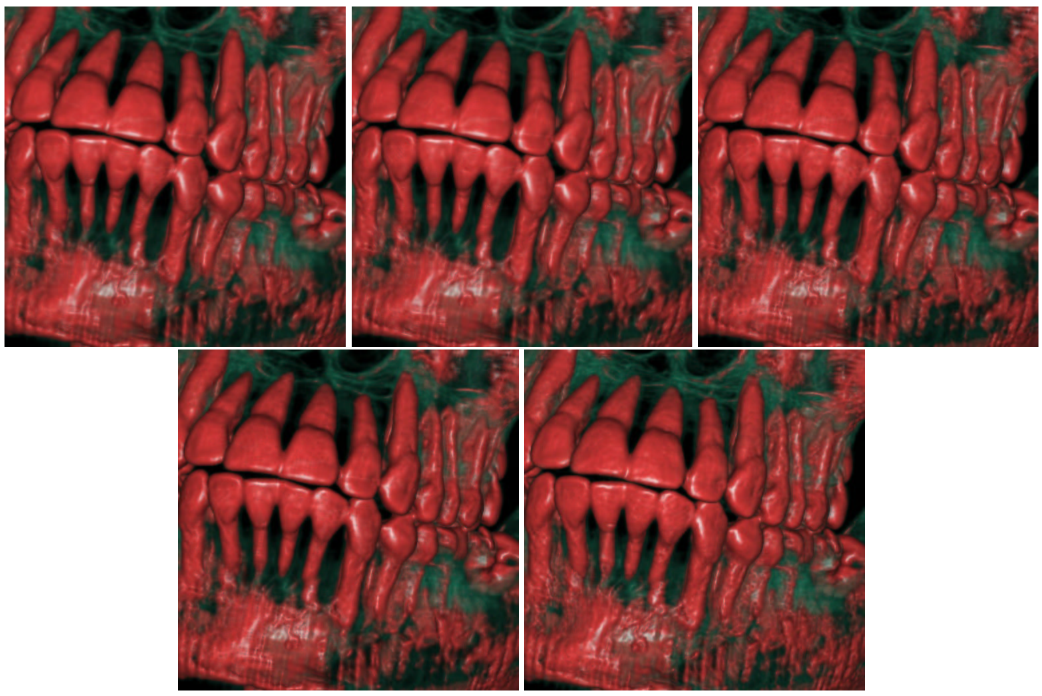

Figure 9: Sheared trilinear, bilinear, barycentric (top row), and trilinear interpolation (bottom left) on the BCC grid and trilinear interpolation on the CC grid (bottom right) for the skull dataset

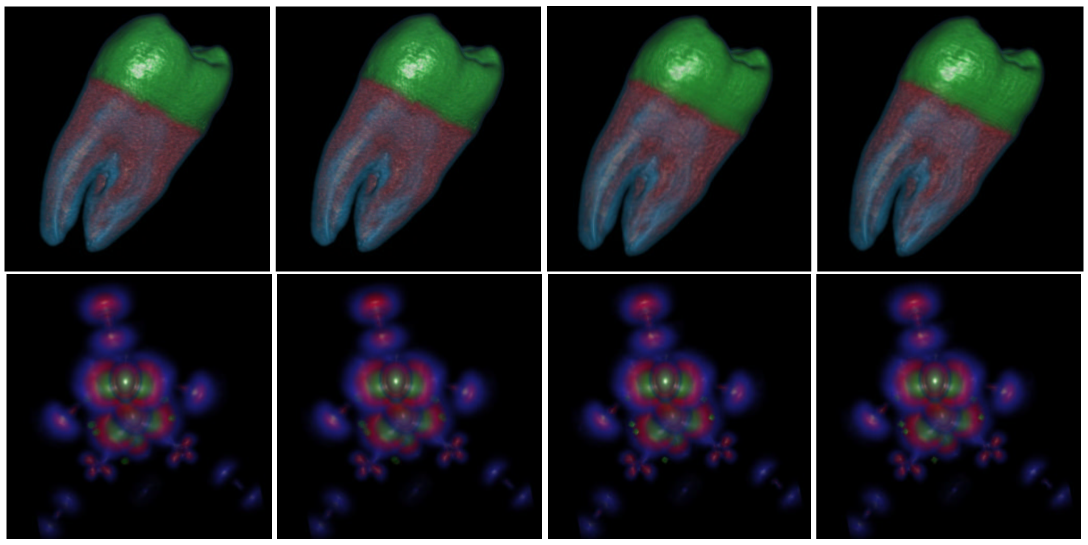

Figure 10: Top row: bilinear, trilinear interpolation on the CC grid (left) and on the BCC grid (right) for the tooth dataset. Bottom row, from left to right: sheared trilinear, bilinear, barycentric, and trilinear interpolation for the hipiph dataset on the BCC grid

Grid	*Interpolator*	*Samplerate*	*RenderingTime*
BCC	bilinear	0.75497	11sec922ms
BCC	barycentric	0.75497	17sec722ms
BCC	sh.trilinear	0.75497	18sec945ms
BCC	trilinear	0.75497	26sec551ms
CC	bilinear	0,53385	20sec172ms
CC	trilinear	0.53385	20sec170ms
CC	trilinear	0.5	24sec718ms
BCC	barycentric	0.5	26sec006ms
BCC	sh.trilinear	0.5	27sec100ms
BCC	trilinear	0.5	38sec959ms
BCC	bilinear	0.37745	26sec822ms
BCC	barycentric	0.37745	33sec747ms
BCC	sh.trilinear	0.37745	35sec499ms
BCC	trilinear	0.37745	50sec822ms
CC	trilinear	0.3	39sec650ms
BCC	barycentric	0.3	41sec963ms
BCC	sh.trilinear	0.3	43sec694ms
BCC	trilinear	0.3	1min3sec268ms

Table 2: Timings for several different interpolators on the BCC and CC grid for the fuel dataset

Grid	*Interpolator*	*Samplerate*	*RenderingTime*
BCC	bilinear	0.73340	1sec842ms
BCC	sh.trilinear	0.73340	2sec383ms
BCC	barycentric	0.73340	2sec416ms
BCC	trilinear	0.73340	3sec098ms
CC	bilinear	0.51885	2sec832ms
CC	trilinear	0.51885	3sec201ms
BCC	sh.trilinear	0.5	3sec149ms
BCC	barycentric	0.5	3sec225ms
CC	trilinear	0.5	3sec300ms
BCC	trilinear	0.5	4sec337ms
BCC	bilinear	0.36670	3sec208ms
BCC	sh.trilinear	0.36670	4sec087ms
BCC	barycentric	0.36670	4sec212ms
BCC	trilinear	0.36670	5sec621ms
BCC	sh.trilinear	0.3	4sec775ms
BCC	barycentric	0.3	4sec948ms
CC	trilinear	0.3	4sec968ms
BCC	trilinear	0.3	6sec752ms

Table 3: Timings for several different interpolators on the BCC and CC grid for the Marschner-Lobb dataset

Grid	*Interpolator*	*Samplerate*	*RenderingTime*
BCC	bilinear	0.75554	7sec999ms
BCC	barycentric	0.75554	11sec648ms
BCC	sh.trilinear	0.75554	12sec325ms
BCC	trilinear	0.75554	16sec941ma
CC	trilinear	0.5	15sec958ms
BCC	barycentric	0.5	17sec134ms
BCC	sh.trilinear	0.5	17sec713ms
BCC	trilinear	0.5	24sec887ms
BCC	bilinear	0.37777	17sec864ms
BCC	barycentric	0.37777	22sec190ms
BCC	sh.trilinear	0.37777	23sec114ms
BCC	trilinear	0.37777	32sec628ms
CC	trilinear	0.3	25sec624ms
BCC	barycentric	0.3	27sec658ms
BCC	sh.trilinear	0.3	28sec576ms
BCC	trilinear	0.3	40sec735ms

Table 4: Timings for several different interpolators on the BCC and CC grid for the hipiph dataset

the BCC grid the resample point lies in. Sheared trilinear interpolation takes advantage of the fact that a BCC grid can be seen as sheared CC grid with sheared cubic cells.

From these four reconstruction schemes, bilinear interpolation is fastest, however, it can exhibit artifacts. The second fastest schemes are barycentric and sheared trilinear interpolation. However, sheared trilinear interpolation achieves qualitatively superior results as compared to the other schemes. Trilinear interpolation is quite slow due to the costly interpolation of additional corner points.

As a conclusion, sheared trilinear interpolation is the method of choice for fast yet high quality reconstruction of BCC grids. Faster reconstruction can be achieved with bilinear interpolation which however is prone to artifacts. Sheared trilinear interpolation in the BCC grid is also comparable in speed and quality to trilinear interpolation in the CC grid.

6 Future Work

For a better understanding of the effects of the presented interpolation schemes within the BCC grid, it is necessary to investigate them in the frequency domain. Especially various methods for designing and evaluating reconstruction filters should be specifically adopted and also applied to spherically symmetric filters for the use in the BCC grid.

By sampling a function on a BCC grid, we assume that that function has a spherical frequency response as opposed to the CC grid where a cubic frequency response is assumed. Consequently, the ideal reconstruction filter for the BCC grid is spherically symmetric. Unfortunately, this ideal reconstruction filter has, similar to the 1D case, infinite extent and is therefore impracticable. However, we can use any filter designed in one dimension (see for example Mitchell and Netravali [17], Marschner and Lobb [15], Möller et al. [18]) through radial extension (i.e., $f(r) = f(\sqrt{x^2+y^2+z^2})$ and convolution with the sample points (see Figure 11). Preliminary results with rotationally symmetric filter interpolation in the BCC grid are quite promising. Figure 12 shows an example for a high quality reconstruction using, in this case, a rotated cubic Catmull-Rom reconstruction filter.

Generally, spherically symmetric filters are not separable. This means that schemes as proposed by Bentum et al. [1] cannot be used. This and the assumed cubic frequency response of CC data is the reason that spherically symmetric filters are rarely used with the

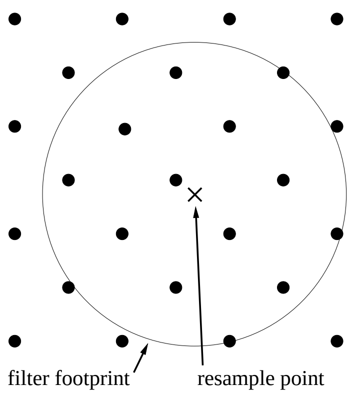

Figure 11: Using a rotational symmetric filter for interpolation by convolving it with the sampling points.

CC grid. However, in case of the BCC grid they have the potential of high quality image generation but with high computational cost.

Further, we would like to investigate other acceleration techniques. In this paper we employed early ray termination and substituting trilinear by bilinear interpolation. Especially space leaping techniques have the potential to significantly speed up the rendering process. Such techniques have not yet been exploited for BCC grids although they have been extensively used for CC grids [14, 20].

Acknowledgments

We thank Klaus Mueller for many fruitful discussions and for letting us redraw Figure 1 from similar figures in his papers [19, 21].

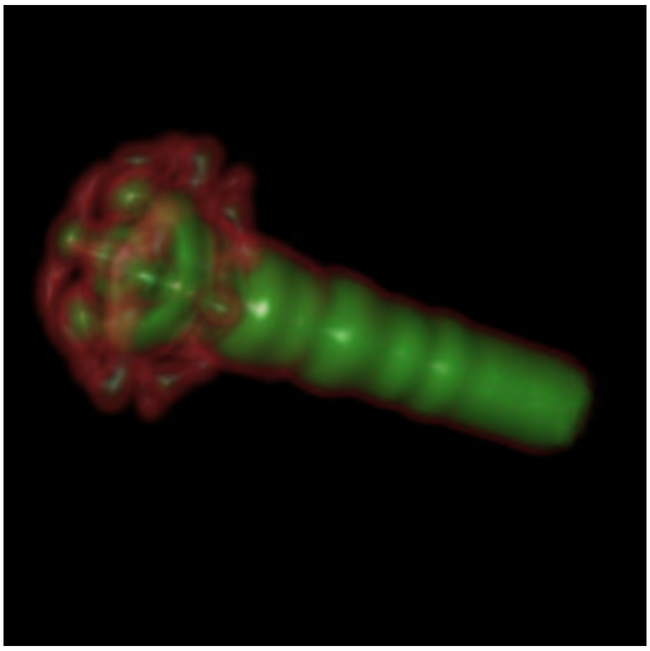

Figure 12: High quality reconstruction of the fuel dataset with a rotated cubic Catmull-Rom reconstruction filter.

References

[1] M. J. Bentum, B. B. A. Lichtenbelt, and T. Malzbender. Frequency analysis of gradient estimators in volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 2(3):242–254, September 1996.

[2] H. Carr, T. Theußl, and T. Möller. Isosurfaces on optimal regular samples. Technical report, University of British Columbia, Vancouver, 2002.

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

[4] D. E. Dudgeon and R. M. Mersereau. *Multidimensional Digital Signal Processing*. Prentice-Hall, Inc., Englewood-Cliffs, NJ, 1st edition, 1984.

[5] T.C. Hales. Cannonballs and honeycombs. *Notices of the AMS*, 47(4):440–449, April 2000.

[6] D. Hilbert. Mathematische Probleme. *Nachrichten der Königlichen Gesellschaft der Wissenschaften zu Göttingen, mathematisch-physikalische Klasse*, 3(1):253–297, 1900.

[7] L. Ibáñez, C.Hamitouche, and C.Roux. Determination of discrete sampling grids with optimal topological and spectral properties. In *Procceedings of the 6th International Workshop in Discrete Geometry for Computer Imagery DGCI'96*, pages 181–192, 1996.

[8] 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.

[9] L. Ibáñez, C. Hamitouche, and C. Roux. Ray-tracing and 3D objects representation in the BCC and FCC grids. In *Lecture Notes in Computer Science 1347*, pages 235–241, 1997.

[10] L. Ibáñez, C. Hamitouche, and C. Roux. A vectorial algorithm for tracing discrete straight lines in *N*-dimensional generalized grids. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):97–108, April/June 2001.

[11] J. Kepler. Strena seu nive sexangula, 1611. Translated by L. L. Whyte as ”The Six-Cornered Snowflake”, 1966 (Oxford Univ. Press).

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

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

[14] M. Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, 9(3):245–261, July 1990.

[15] S. R. Marschner and R. J. Lobb. An evaluation of reconstruction filters for volume rendering. In R. Daniel Bergeron and Arie E. Kaufman, editors, *Proceedings of Visualization '94*, pages 100–107. IEEE, October 1994.

[16] M. Meißner, J. Huang, D. Bartz, K. Mueller, and R. Crawfis. A practical evaluation of popular volume rendering algorithms. In *Proceedings of the 2000 IEEE Symposium on Volume Visualization*, pages 81–90, 2000.

[17] D. P. Mitchell and A. N. Netravali. Reconstruction filters in computer graphics. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):221–228, October 1988.

[18] T. Möller, R. Machiraju, K. Mueller, and Roni Yagel. Evaluation and design of filters using a Taylor series expansion. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):184–199, April/June 1997.

[19] N. Neophytou and K. Mueller. Space-time points: Splatting in 4D. In *Symposium on Volume Visualization and Graphics*, pages 97–106, Boston, MA, October 2002.

[20] M. Šrámek. *Visualization of volumetric data by ray tracing*. PhD thesis, Vienna University of Technology, 1996.

[21] J. Sweeney and K. Mueller. Shear-warp deluxe: The shear-warp algorithm revisited. In *Joint Eurographics - IEEE TCVG Symposium on Visualization (VisSym '02)*, pages 95–104, Barcelona, Spain, May 2002.

[22] T. Theußl, T. Möller, and M. E. Gröller. Optimal regular volume sampling. In *IEEE Visualization '01 (VIS '01)*, pages 91–98, Washington - Brussels - Tokyo, October 2001. IEEE.

[23] G. M. Treece, R. W. Prager, and A. H. Gee. Regularised marching tetrahedra: improved iso-surface extraction. *Computers and Graphics*, 23(4):583–598, August 1999.

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