

# VDVR: Verifiable Visualization of Projection-Based Data

Ziyi Zheng, Wei Xu, and Klaus Mueller, *Senior Member, IEEE*

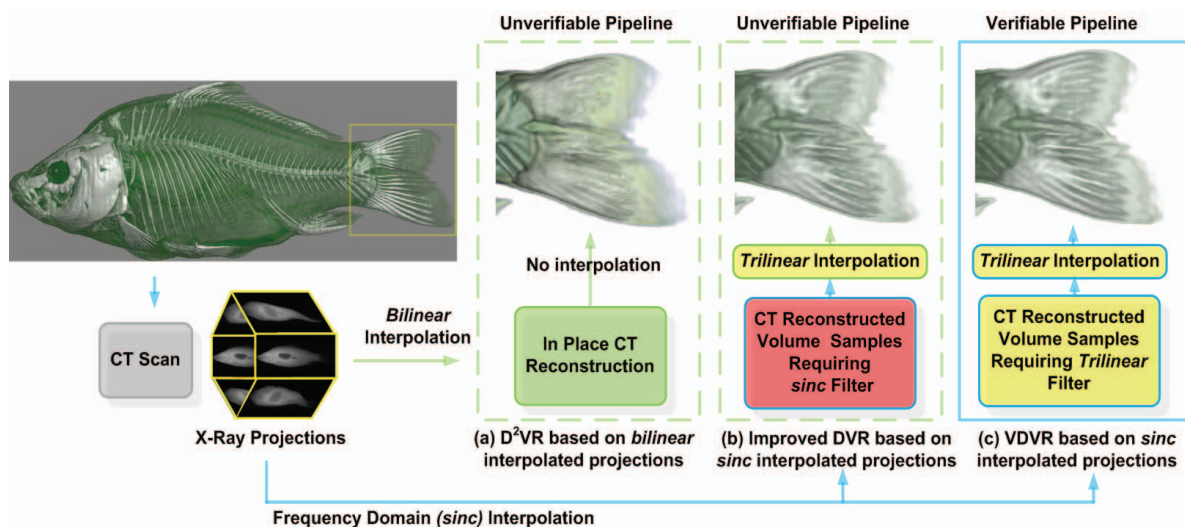


Fig. 1. CT data acquisition, reconstruction and visualization pipelines for (a)  $D^2VR$  [29], (b) unverifiable, (c) our verifiable framework.

**Abstract**—Practical volume visualization pipelines are never without compromises and errors. A delicate and often-studied component is the interpolation of off-grid samples, where aliasing can lead to misleading artifacts and blurring, potentially hiding fine details of critical importance. The verifiable visualization framework we describe aims to account for these errors directly in the volume generation stage, and we specifically target volumetric data obtained via computed tomography (CT) reconstruction. In this case the raw data are the X-ray projections obtained from the scanner and the volume data generation process is the CT algorithm. Our framework informs the CT reconstruction process of the specific filter intended for interpolation in the subsequent visualization process, and this in turn ensures an accurate interpolation there at a set tolerance. Here, we focus on fast trilinear interpolation in conjunction with an octree-type mixed resolution volume representation without T-junctions. Efficient rendering is achieved by a space-efficient and locality-optimized representation, which can straightforwardly exploit fast fixed-function pipelines on GPUs.

**Index Terms**—Direct volume rendering, computed tomography, filtered back-projection, verifiable visualization.

## 1 INTRODUCTION

Scientists and practitioners frequently utilize visualization tools for diagnosis and decision-making. One application is the visualization of volume datasets derived from medical and industrial *computed tomography (CT)*. CT acquires X-ray projection data of an object or patient from different vantage points and subsequently employs *CT reconstruction* to produce these volume data sets. Via direct volume rendering (DVR), users can then look for fine details such as hairline fractures, small pathological features such as tumors, and textures of diagnostic value. To be useful, the volume visualization tools must form a reliable basis for their judgment. Additionally, users also require interactive rendering speed to freely examine the data.

Conceptually, we may perceive CT reconstruction as a *transformation* of the acquired raw projection data into a spatially coherent format to make this analysis and interaction more direct and thus more efficient and effective. Since CT scanning potentially acquires redundant data, the spatial pooling of these data within the CT reconstruction can also be seen as producing a *compression* of

the raw projection data. Our paper addresses the need for a loss-bounded transformation and compression in volume visualization.

Visualization researchers, as well as the users, either know or at least suspect that their visualization tools may not be completely truthful to the underlying data. This is often rooted in compromises that need to be made for balancing rendering speed and quality. Much research has focused on recovering fine details from volumetric data by designing and using interpolation filters of higher quality. These filters are mostly based on theoretical derivations and do not consider the origin of the data themselves. However, as is the case for CT, the volumetric data subject to interpolation in the rendering stage are often not the actual raw data, but only derived from them. Thus, any quality-enhancing effort neglecting this transformation stage cannot guarantee verifiable results.

As motivated above, for CT data the raw scalar field is naturally available from the CT acquisition process and therefore a verifiable visualization pipeline must integrate the rendering stages with the scalar field generation. The recent insightful volume rendering approach by Rauterk et al. [29] recognized this important relationship. Their algorithm, termed Direct DVR ( $D^2VR$ ), integrates the raw data transformation (the CT reconstruction) stage and the rendering stage by directly generating the samples required for rendering from the raw data (X-ray projections) in place. In other words, they generate a transient volume dataset that does not require any further interpolation in volume space. They convincingly show that this has great potential for improving rendering quality. Similarly, the CT

• Ziyi Zheng, Wei Xu and Klaus Mueller are with the Center of Visual Computing, Stony Brook University,

E-Mail: {zizhen, wxu, mueller}@cs.sunysb.edu.

Manuscript received 31 March 2010; accepted 1 August 2010; posted online 24 October 2010; mailed on 16 October 2010.

For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

community also noticed these resampling issues that motivated D<sup>2</sup>VR, suggesting the same direct pipeline [16]. Yet, as we will argue, there is still an overlooked compromise in D<sup>2</sup>VR which degrades it into an unverifiable data visualization pipeline.

As past efforts show, directly visualizing CT raw data is an expensive operation, since it neglects the inherent advantages of CT reconstruction, that is, the spatial coherence and data compression it provides. First, for standard DVR to render an image of  $n^2$  pixels one requires  $O(n^3)$  off-grid sample interpolations in volume space (to generate the densities along the rendering rays). On the other hand, assuming  $O(n)$  projections, for D<sup>2</sup>VR to generate  $O(n^3)$  volume samples one requires  $O(n^4)$  projection data interpolations. Translating these complexity arguments into practice, this causes D<sup>2</sup>VR to be about 50 times slower than DVR, assuming tri- and bi-linear interpolation for DVR and D<sup>2</sup>VR respectively. In fact, this led the CT D<sup>2</sup>VR researchers to only develop a real-time 2D slice-viewer, shying away from volume rendering arguing the lack of sufficient computing power. Second, the prior CT reconstruction of a volume also avoids the poor locality of the projection data when mapping spatially coherent data access requests. The texture fetching pattern of D<sup>2</sup>VR in volume rendering is a *sine* function (also called the *sinogram*) which does not map well into a GPU rendering pipeline. We can observe this from the results obtained by Xu and Mueller [34]. Their GPU-accelerated D<sup>2</sup>VR only achieved speedups of 1.3-1.5 after applying a number of (albeit standard) acceleration schemes, such as occlusion culling and empty-space skipping.

In order to bring verifiable visualization into practice, we propose a framework we call *Verifiable DVR*, or *VDVR*. Our verification procedure bridges the currently existing disconnect between the raw projection data and their visualization via volume rendering. It helps to make visualizations verifiable since we guarantee a preset error tolerance that is applied in the CT reconstruction step.

The fundamental difference between D<sup>2</sup>VR, standard DVR and our VDVR is illustrated in Fig. 1. Apart from the nature of the interpolated data (raw projection data for D<sup>2</sup>VR, volume data for DVR and VDVR), the differences stem from the filter used for the interpolation. D<sup>2</sup>VR, shown in panel (a), interpolates the projection data using a bilinear filter rather than the ideal *sinc* filter. Thus, although D<sup>2</sup>VR effectively eliminates sampling errors in the volume domain, it still commits errors in the projections domain, which are not explicitly verified (we will show later that such a verification would lead to a prohibitively inefficient D<sup>2</sup>VR algorithm). We can observe the resulting fidelity losses especially for the fine details on the fish tail. Conversely, both standard DVR and VDVR can effectively use *sinc*-interpolated projections, since the CT reconstruction is only a one-time process. However, volumes used in standard DVR typically are not generated with the sample interpolation errors in mind, and so they must use an ideal *sinc* filter to make guarantees on accuracy. Therefore, when a more practical trilinear interpolation filter is used instead, fine details cannot be preserved which is evidenced in panel (b). On the other hand, a renderer presented with a VDVR-certified volume may safely use the interpolation filter for which the volume has been verified (we demonstrate this with the trilinear filter to show the gains in speed that can be obtained). This enables all details to be recovered in the rendering, as is evidenced in panel (c).

Our paper is structured as follows. In Section 2 we discuss related works. Section 3 provides the theoretical derivations of our research, while Section 4 discusses practical issues. Section 5 presents implementation aspects. Section 6 shows results, Section 7 offers some discussions, and Section 8 presents conclusions.

## 2 RELATED WORK

Volume rendering includes a wide range of techniques. Engel et al. give an overview of real-time volume rendering methods [10]. Most approaches focus on overcoming discretized representations to recover a smooth signal. This also includes high-quality normal vector reconstruction. These methods treat volume grids rather than

scalar fields as raw data, which do not conform to the notion of “verifiable visualizations” as proposed by Kirby and Silva [19]. Kirby and Silva considered the errors committed within the visualization process in the data generation process, using flow simulation as an example. They also later applied their verifiable visualization to iso-surface extraction [13]. Our work proposes a similar verifiable framework but we focus on volume rendering and target one of the prominent sources of volumetric datasets, CT scanners and CT reconstruction. In their work, Kirby and Silva proposed convergence tests for benchmarking different visualization methods. In our case we perform evaluations via Root-Mean-Square (RMS) error between the scanned and the reconstructed object.

In volume visualization, off-grid sample points are typically estimated via interpolation<sup>1</sup>. There has been a great deal of work on interpolation and filters in the volume rendering community [25] [32][11][4] and a variety of trade-offs have been identified. A major finding in this regard is that filters with good anti-aliasing properties do exist but these (i) tend to be expensive to evaluate and (ii) also tend to suppress frequencies in the higher portions of the pass-band more than inexpensive low-quality filters, smoothing the results.

To cope with the latter shortcoming, recent work [6] proposed to pre-filter the data first, using deconvolution in the frequency domain, to then produce the undistorted function at interpolation time. Formalizing the pre-filtering as a projection of the acquired signal into the space of band-limited signals has been the subject of a paper by Unser [33]. In this theoretical framework the error is predicted by averaging the amplitudes of the signal’s frequency spectrum, which however is, as we will show, a rather conservative bound. We find that by considering the spectrum of phase-shifts as well, via a curvature-based metric, the error bound can be made significantly tighter without loss in signal fidelity, and this greatly reduces the number of required grid points for a given verifiable accuracy.

To deal with the other factor, filter evaluation cost, recent work by Csébfalvi and Domonkos [7] has proposed (uniform) upsampling as a means to facilitate the use of less expensive interpolation filters without impairing rendering quality. They show that an upsampling rate of 2 is roughly sufficient to reduce the filter complexity from cubic to linear and so gain an order of magnitude speedup in GPU. We aim for a similar goal but add grid points adaptively and not uniformly. Specifically, but without loss of generality, we focus on the trilinear interpolation filter to take advantage of the hardwired support offered by fixed-function GPU pipelines.

In this context, we would like to distinguish between *upsampling* and *oversampling*. The former occurs as a post-process on existing volume data with the goal of increasing their resolution (see above), and an efficient method is to zero-pad the volume data in the frequency domain (as was done in [7]). Conversely, we attack the data fidelity problem at its root, that is, during volume data *generation*, and at that stage oversample the volume data in areas with critical detail. In other words, we sample the (potentially continuous) function being reconstructed at a higher rate to represent all detail (under the constraint of the interpolation filter used later on).

The mixed-resolution representation we devise has a fundamentally different goal than octree-based multi-resolution frameworks, such as [20][12][23]. For example, the Gigavoxel framework [5] can provide interactive rendering of massive volume data. Aimed at providing aliasing-free level-of-detail (LOD), most of these multi-resolution approaches store multi-resolution data in different textures and render each brick individually. The high resolution of their input also hides, to some extent, aliasing issues [5]. Similarly, in the physics simulation domain, AMR is a refinement structure which was first developed by Kähler et al. [17] [18] and further improved by Marchesin et al [27]. In contrast to these methods, our representation data is directly derived from the raw projection data and can so extend resolution only when needed to

<sup>1</sup> One may consider this sample *interpolation* also a reconstruction of the continuous signal at the sample’s location. We use the term ‘interpolation’ since ‘reconstruction’ is already used in the context of CT reconstruction.

preserve detail. Therefore, our mixed-resolution framework is more adaptive and less brute force when it comes to data storage. In addition, we do not aim for a multi-resolution representation that can provide smooth transitions from low-resolution to high-resolution. Rather, we only keep the leaves of the octree, at the local level that preserves the fidelity of the (transformed) raw data.

T-intersections occur when two boundaries of different resolutions meet. They can cause visible banding artifacts if not handled properly. The approach by Ljung et al. [24] smoothly interpolates between these mixed-resolution boundaries, and Beyer et al. [2] introduce a scheme that blends samples nearby. This approach does not give explicit error control and therefore it is not verified by the raw data. Also, because the refinement cell position is grouped into a coarse granular octree (only 2 levels), it is inefficient to handle sparse refinement regions (void region and thin structure). In contrast, our data method can directly account for visualization errors and support finer granularity.

Finally, we note that noise in the projection data has a definite influence on the complexity of the volumes our framework will generate. The more noise in the raw data exists the more (albeit false) detail would be reconstructed. We assume that noise (or better, what has been identified as noise) is removed from the raw data either in a pore-processing step or during the CT reconstruction itself. Proven methods (for example, [21][36]) exist for either of these strategies and so this does not represent an obstacle for our framework. Also, in practice detector binning is often applied to remove noise and aliasing, in particular in low-energy imaging scenarios.

### 3 THEORETICAL CONSIDERATIONS

#### 3.1 Background: CT Reconstruction

Here we provide some background on CT reconstruction and use the 2D case for ease of illustration (pictured in Fig. 2). Any point within the circle defined by the intersection of all detector shadows can be accounted for in the X-ray projections and later be reconstructed via filtered back-projection (FBP). The back-projection is essentially inverse (X-ray) volume rendering, that is, a volume point is calculated by summing the contributions of all rays that emanate from the corresponding projection pixels and pass through the point. For parallel-beam projection geometries, the Fourier transform for each projection constitutes a radial slice of the imaged signal's Fourier spectrum, as shown in Fig. 2b for the amplitude portion. This spectrum is available on a polar grid and is bandlimited to  $\pm N/2$ , where  $N$  is the projection's resolution (the number of pixels). This existence of a bandlimit will play an important role in our error analysis (more on this later). Also note that the outer circle areas are less tightly sampled. This can be compensated for by pre-filtering the data by a ramp (Ram-Lak) filter, but we might also use more noise-suppressing filters, such as the Shepp-Logan filter [31], which multiply the ramp by a window to de-emphasize high frequencies.

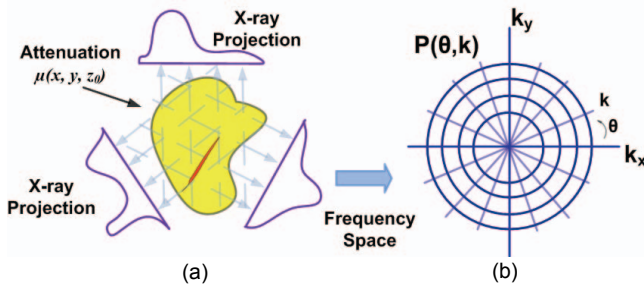


Fig. 2. The X-ray transform and its Fourier spectrum in 2D.

The (analytical) 2D filtered back-projection formula is:

$$f(x, y) = \int_0^\pi \int_{-\infty}^{+\infty} P_\theta(\omega) |\omega| e^{j2\pi\omega x} d\omega d\theta = \int_0^\pi q_\theta(t) dt \quad (1)$$

where  $t = x\cos\theta + y\sin\theta$  is the projected location of point  $(x, y)$  and  $\theta$  is the projection angle.  $P_\theta(\omega)$  is the 1D Fourier transform of a

1D projection  $p_\theta(t)$ . The inner integral is the 1D inverse Fourier transform resulting in  $q_\theta(t)$ . The outer integral represents the summed back-projection of all  $q_\theta(t)$ .

In practice,

$$f(x, y) \approx \frac{\pi}{K} \sum_{i=1}^K q_{\theta_i}(x\cos\theta_i + y\sin\theta_i) \quad (2)$$

where  $q_{\theta_i}(t)$  are the ramp-filtered projections. Eq. (1) and (2) formalize the reconstruction pipeline. For each 1D projection  $p_{\theta_i}(t)$ , we first apply a 1D Fourier transform to yield  $P_{\theta_i}(\omega)$ . Each  $P_{\theta_i}(\omega)$  is ramp filtered and then an inverse Fourier transform computes  $q_{\theta_i}(t)$  (for 2D data the ramp filtering needs to be done along the projection rows in 1D). Finally, we sum all back-projected  $q_{\theta_i}(t)$  at the voxels and divide this sum by the number of projections  $K$ . This  $K$  should be at least  $(\pi/2)N$  such that the polar Fourier transform has about the same resolution in  $\theta$  at the periphery and  $k$  (see Fig. 2b) [31].

#### 3.2 Frequency Domain Projection Upsampling

In FBP, after ramp-filtering, the projections are stored as discrete samples, to be interpolated in the later back-projection stage. Here, bilinear interpolation is mostly used in GPU-based CT reconstruction for its fast speed performance. But as discussed in Section 2, inexpensive filters cause artifacts which we would like to avoid.

The general mindset of our approach is to provide a sufficient amount of up/oversampling to allow for a verifiable approximation of the underlying continuous function by a piecewise linear function, which we can then interpolate by means of a linear filter. This mindset applies to both the projection domain and later to the volume domain. Our first task is to provide such a faithful upsampling for the projection data. As mentioned, frequency domain upsampling is the most appropriate solution for this. It is equivalent to using an ideal *sinc* filter, but without the high cost of its infinite support. In a 2D Fourier transform, a signal given in frequency space can be up-sampled by padding zeros at both ends of the spectrum (used, for example, in [26] to improve the quality of Fourier volume rendering). One can then convert the obtained signal back into signal space at the new (higher) resolution. A potential issue in frequency domain-based upsampling are sharp boundaries that may exist in the signal. Discontinuities at these boundaries introduce high frequencies. These high frequencies can be avoided by a mirror extension, which ensures a smooth transition at the boundaries [1]. Alternatively, one can also use spatial-domain windowing [22].

More concretely, our goal is a high-quality upsampling of the ramp-filtered data. Let us denote the signal as  $S$ , the upsampling operator as  $U$  and the ramp filter as  $F$ . Since the operators are linear we may either perform  $S \otimes U \otimes F$  or  $S \otimes F \otimes U$ . If we use filters other than the *sinc* either option will contain strong aliasing, since the affected high frequency bands are magnified by the high-pass filtering of  $F$ . It is straightforward to either integrate on-the-fly *sinc*-interpolation or prior frequency domain upsampling into the VDVR pipeline. It merely forms a pre-processing stage and the additional computation or samples will only be needed in the CT reconstruction. On the other hand,  $D^2VR$  cannot incorporate on-the-fly *sinc*-filtering or pre-computed upsampling easily. For the former the computational overhead would be prohibitive, and for the latter a  $64\times$  storage increase in GPU memory (for  $8\times$  upsampling) would be challenging. Therefore they do not perform any projection upsampling.

To illustrate the need for frequency domain upsampling, we performed tests on a challenging dataset, the Marschner-Lobb function [27]. Fig. 3 shows the RMS error (RMSE) behavior of filtered back-projection for projections obtained from the analytical ML function (also in Figs. 11-13 and 15) set within a  $(64/\sqrt{2})^3$  cube. We observe that the RMS error is still decreasing after  $64\pi/2 \approx 72$  projections, which is the theoretical minimum number of projections needed (see Section 3.1). Further, since the ML function is set into a cube, these sharp boundaries extend its frequency space to infinity, causing Gibbs phenomenon on the cube boundary which would interfere with the measurement. Therefore we

perform the test in the cube interior only (87.5% of the cube length). The RMSE for 72 projections is 0.7% (blue curve). In contrast, the red curve shows that the RMSE for sampling used in D<sup>2</sup>VR stops improving after 30 projections, at about 5% RMSE. We therefore conclude that frequency-based (or *sinc*-based) upsampling is clearly needed to create a verifiable result.

The upsampling process is plugged into the CT reconstruction pipeline as follows. The inputs are the 2D X-ray projections obtained from the CT scanner (or obtained with a high-quality raycaster used in our simulations). For each projection, a 2D FFT is obtained, zero-padded, and a 2D inverse FFT is run. Following, we perform a 1D FFT for each line, ramp-filter, and do a 1D inverse FFT.

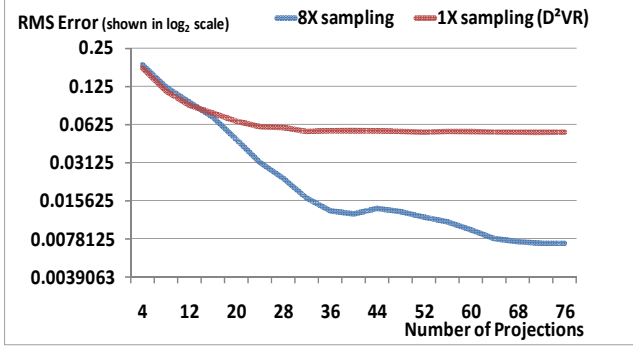


Fig. 3. Reconstruction error of filtered back projection methods. Volume registration is done by normalizing standard deviation and sample mean. Measurements are based on 448×448×512 samples.

### 3.3 Interpolation Error Assessment

#### 3.3.1 Error assessment for the Linear Filter

For the linear filter the largest error occurs at the local peak or valley where the maximum curvature is located [30]. Fig. 4a illustrates this scenario for a single frequency, where the largest error occurs around the *sine* function's peak. Given the sampling distance  $d$ , the max absolute error  $E_i$  for a specific wavelength  $T_i$  with amplitude  $A_i$  is:

$$E_i = |A_i| \left(1 - \sin\left(\frac{2\pi}{T_i} \left(\frac{1}{4}T_i - \frac{d}{2}\right)\right)\right) \quad d < T_i/2 \quad (3)$$

where the maximum interpolation error  $E_i$  is a function of the sampling distance  $d$  and the signal period  $T_i$ . The distance  $d$  and period  $T_i$  are connected by the oversampling rate. If  $d = T_i/2$ , the sampling rate is just below the Nyquist sampling rate. In this case, the maximum error for linear interpolation could be 100% of the *sine* peak value. If  $d = T_i/16$  (equivalent to an 8× oversampling rate), this will guarantee that the error is less than 1.92% of the maximum (peak) value. Fig. 4b illustrates the error as a function of oversampling rate and signal frequency. The error decreases with increasing oversampling rate and/or decreasing signal frequency.

Of course, a signal is a composite of multiple frequencies and therefore these errors would possibly compound. However, most likely these frequencies would be phase shifted which would reduce

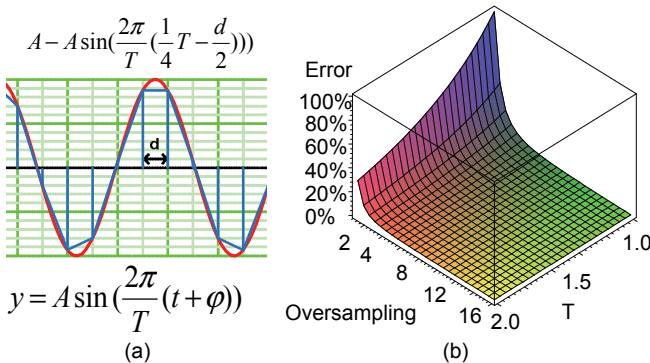


Fig. 4. *Sine* signal reconstructed with linear interpolation and its error.

the local curvature and thus alleviate the error. Given a set of  $A_i$ , the largest error occurs when all *sine* peaks accumulate in one point. The largest error for this composite signal is:

$$E_{\max} \leq \sum_{i=0}^{N-1} |A_i| \max\left\{\frac{E_i}{|A_i|}\right\} = \sum_{i=0}^{N-1} |A_i| \left(1 - \sin\left(\frac{2\pi}{T_i} \left(\frac{1}{4}T_i - \frac{d}{2}\right)\right)\right) \quad (4)$$

where  $N$  is the length of the signal which is also the number of frequencies obtained with the FFT.

We shall now give a general impression on what this means in practice. In Fig. 5a the blue curve shows the frequency amplitude for the central line of the X-ray projection of the carp dataset, while the red curve shows these amplitudes after ramp-filtering. For the blue curve, we see that the highest amplitudes are located at the low frequencies (left). The panel (b) shows the errors as a function of frequency and oversampling. We observe that for 1× resolution the highest error (100%) is located at the highest frequencies and falls off according to Eq. (3), while for 8× resolution the errors all stay below 2%. Next, panel (c) shows the product of the amplitudes (the blue curve) and the error map. The plot shows the error at the traditional resolution on a scale from 0 to 0.006 (1.0 is the maximum). Summing the errors for the full amplitude spectrum will amount to 22% of the maximum scalar value of the carp. Conversely, the errors for the 8× oversampling case are reduced by almost two orders of magnitude, and their sum is 0.4% of the maximum scalar value. We thus conclude that the residual's overall impact is negligible. Furthermore, for the filtered signal (red curve), the sum is even less.

As mentioned, the amplitude-based theoretical error assuming the worst phase shift is too conservative and likely impractical to use. Another error can be derived by taking phase shift into consideration. Let  $M_2$  be the maximum absolute value of the curvature. Then the maximum error for the sampling distance  $d$  is

$$E_{\max} \leq \frac{M_2}{2} \left(\frac{d}{2}\right)^2 = \frac{M_2}{8} d^2 \quad (5)$$

The proof [30] of Eq. (5) is based on Taylor's Theorem. In general, the amplitude-based error bound is tighter for single frequency signals and the curvature-based error bound is tighter for compound frequency signals. We can use both to estimate the error.

#### 3.3.2 Error assessment for the Bilinear Filter

For the 2D case, the amplitude-based error bound is

$$E_{\max} \leq \sum_{j=0}^{M-1} \sum_{i=0}^{N-1} |A_{i,j}| \max\left\{\frac{E_{i,j}}{|A_{i,j}|}\right\} \quad (6)$$

where  $N$  and  $M$  are the width and height of the signal as a 2D image. Since there is no straightforward analytical solution, we computed the resulting 2D percentage-error map by extensive exhaustive search. For a given frequency  $f_{i,j}$ , we generated 2D *sine* waves with a uniform distribution of phase shifts  $\varphi$  and then find the maximum error inside a bilinearly interpolated unit square.

Fig. 5(d-f) show the results for this 2D analysis, for a 2D X-ray projection of the carp dataset. We observe similar effects than in the 1D case. Panel (d) shows the image's amplitude spectrum using a log-scaled colormap. Panel (e) shows the error for the regular 1× sampling case (top) and the 8× oversampling case (bottom). Finally, panel (f) shows the errors multiplied by the amplitude spectrum. We observe that 8× oversampling removes the error almost completely.

An important observation we can make in Fig. 5e is that the 8× sampling error map is a direct copy of the 1/8×1/8 center square in the 1× sampling error map (see the illustration linking the top and bottom plots of panel (e)). So the higher degree of oversampling, the more one zooms into the 1× error map. Also, since the raw data (the sinogram) is bandlimited, the amplitude spectrum of the reconstructed signal stays constant but the overall spectrum bandwidth grows at the rate of the oversampling. Both of these facts have important implications for the maximally needed resolution of the verifiable grid, as we shall see soon.

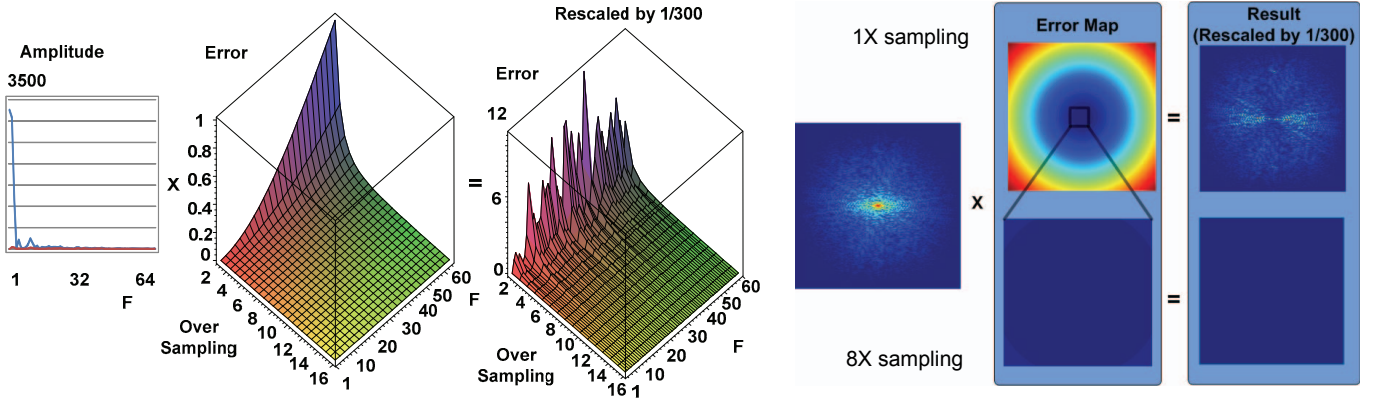


Fig. 5. Linear interpolation error for oversampling. (a-c) show the largest errors for frequencies in 1D and (d-f) show the largest error for frequencies in 2D. (a) shows one half of the spectrum obtained by FFT since the other half is symmetric.

The curvature-based error bound can be derived similarly for the bilinear filter. We use this notation later although it is not exactly the definition of curvature. The paper's supplement document proves an error bound based on the local second order Taylor expansion which is similar to [30][8]. Assuming the signal  $f$  is of class  $C^3$ , let  $M_{x^2}$ ,  $M_{y^2}$ ,  $M_{x^2y}$  and  $M_{xy^2}$  be the largest absolute values of  $f_{x^2}$ ,  $f_{y^2}$ ,  $f_{x^2y}$  and  $f_{xy^2}$  respectively. An error bound for the sampling distance  $d$  is

$$E_{\max} \leq \frac{d^2}{8} (M_{x^2} + M_{y^2}) + \frac{d^3}{4} (M_{x^2y} + M_{xy^2}) \quad (7)$$

We can use this error bound in similar ways as in the 1D case.

### 3.3.3 Error assessment for the Trilinear Filter

If  $N$ ,  $M$  and  $L$  are the size of the 3D signal, the maximum error is

$$E_{\max} \leq \sum_{k=0}^{L-1} \sum_{j=0}^{M-1} \sum_{i=0}^{N-1} |A_{i,j,k}| \max \left\{ \frac{E_{i,j,k}}{|A_{i,j,k}|} \right\} \quad (8)$$

Similarly to the 2D case, we compute the 3D error map in 3D by exhaustive search. For a *sine* function with frequency  $f_{i,j,k}$ , we test a uniform distribution of phase shifts and measure for each the error inside a trilinear interpolated unit cube. Also similar to the 2D case, the oversampling error map can be obtained by extracting the center cube of the standard  $1 \times$  error map volume and expanding it. Then given all  $A_{i,j,k}$  (the 3D amplitude spectrum), we multiply this spectrum by the error volume and compute the sum, which will give us the error bound at the worse phase shift constellation.

For the trilinear filter, the curvature-based error bound can also be derived, as detailed in the paper's supplement. If  $M_\alpha$  is the max-absolute value of  $f_\alpha$  and  $\alpha \in \{x^2, y^2, z^2, x^2y, xy^2, y^2z, yz^2, x^2z, xz^2, xyz\}$ , an error bound for the sampling distance  $d$  is

$$E_{\max} \leq \frac{d^2}{8} (M_{x^2} + M_{y^2} + M_{z^2}) + \frac{d^3}{4} (M_{x^2y} + M_{xy^2} + M_{y^2z} + M_{yz^2} + M_{x^2z} + M_{xz^2} + 3M_{xyz}) \quad (9)$$

## 3.4 Error Control for the Verification

Our aim is use linear interpolation within our verifiable visualization framework and use FBP for CT reconstruction. This requires us to formally quantify the errors incurred with FBP using bilinear or trilinear interpolations (note that other constellations are possible but would have to be formally evaluated as well). There are two sources of error: (i) the interpolation of the 2D projection (raw) data during CT reconstruction of the verifiable volume and (ii) the interpolation of this volume during rendering. We discuss these two errors next.

**Error control in 2D projection interpolation:** In FBP, the ramp-filtering is performed in the frequency domain so no error is incurred at this stage. Then, following Eq. (2), the filtered projections are interpolated and the values summed, multiplied by  $\pi$

and divided by the number of projections  $K$ . Directly applying the bilinear interpolation error on the filtered projections will give us  $K$  errors (one error per projection). However, we are interested in how these errors are reflected in the 3D scalar field. Essentially, this error bound is the sum of all maximum errors for the  $K$  projections, multiplied (normalized) by  $\pi/K$ .

**Error control in 3D volume interpolation:** After FBP, the reconstructed 3D scalar field is sampled and stored as a volume array of discrete samples. Thanks to the important fact that the 3D signal is band-limited we can perform the analysis outlined in Section 3.3.2 on a volume reconstructed at Nyquist resolution ( $1 \times$  oversampling). Based on this reconstruction we can then estimate the error bound for any oversampling rate, for both amplitude and curvature-based error.

In the carp dataset, assuming  $8 \times$  sampling, the amplitude-based bound is 0.4 and the curvature-based error bound is 0.01 (1 is the maximum scalar value). Note that even the curvature based error bound is an over-estimation. As shown in Fig. 3 above, we can get the reconstruction error very close to the original data (0.8% RMSE) even for a challenging dataset, if we first use  $8 \times$  oversampling.

## 4 PRACTICAL CONSIDERATIONS

The entire verifiable pipeline is shown in Fig. 6. The inputs are the X-ray projections and an error threshold  $\epsilon$ . We first determine the upsampling rate of the filtered projections by estimating the error bound. This analysis is based on the projections after ramp-filtering. Then we run frequency domain upsampling according to the verified upsampling rate and ramp-filter the upsampled projections. Following, we perform a CT reconstruction at the Nyquist resolution ( $1 \times$  up-sampling), perform the error analysis and determine the  $\epsilon$ -

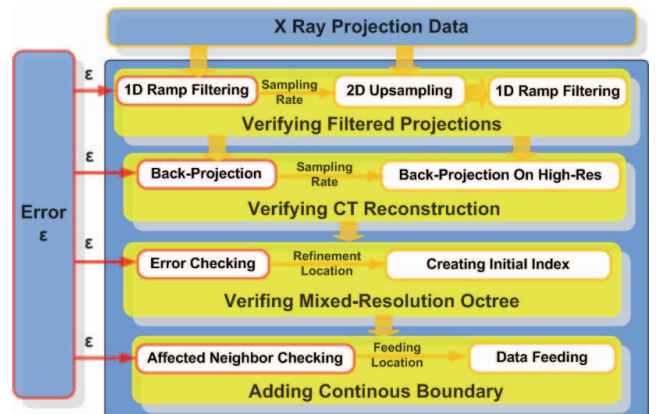


Fig. 6. VDVR pipeline.

verified oversampling rate for the 3D volume. Then we perform back-projection again but now on a high-resolution grid which captures all possible details. We call this the *gold-standard*. To keep within the memory limit, we generate the gold standard in blocks of multiple cells. Within each such block, and from the gold-standard, we then build the mixed-resolution representation only keeping the detail needed. Starting from the typical base resolution commonly used, we classify those cells as subdivision cells which contain finer details. These cells are then represented with more data points. Finally, any potential T-junctions in the mixed-resolution data are removed. In the following, we describe this process in more detail.

#### 4.1 Determining the Verification Parameters

**Verifying the projections:** For the amplitude-based error bound, the maximum error for a filtered projection is the sum of its frequency errors. The final reconstruction error then multiplied by  $\pi/K$  as noted in Section 4.4. We compute the curvature-based error bound in the frequency domain as well. Taking the derivatives of a signal in the spatial domain corresponds to multiplying it by a unit ramp function (the radial frequency  $j\omega$ ) in the frequency domain [3]. Thus, if the amplitude spectrum is multiplied by  $j\omega$  the IFFT will reconstruct the analytical derivative at the grid points. We use this approach to compute a set of images, one for each derivative specified in Eq. (7). We then find the maximum values in each and compute the error according to Eq. (7). The reconstruction error is then the sum of the  $K$  errors for the  $K$  filtered projections, multiplied by  $\pi/K$ .

If both error bounds are larger than  $\epsilon$ , this means we need to increase the upsampling rate. For the amplitude based error, we replace the error map with the one for a 2-times higher sampling rate. The curvature-based error can be simply re-evaluated. We continue this iterative process until the error is below  $\epsilon$ .

**Verifying the gold-standard volume:** The error of the gold-standard volume can be estimated also by ways of these two methods. Both use back-projection to reconstruct a volume at the traditional resolution. For the amplitude-based error bound, we take a 3D FFT, multiply the spectrum by the 3D error map for a certain oversampling rate and sum the errors. For the curvature-based error bound, we use Eq. (9) and estimate the maximum derivatives by taking derivatives in frequency space. Note that here the error bounds are already in 3D space and there is no  $\pi/K$  factor involved. With these two error bounds, the resolution of the gold-standard can be determined.

#### 4.2 Building the Mixed-Resolution Grid

To enable verifiable visualization with efficient hardware-accelerated trilinear filtering (instead of the ideal *sinc* filter or higher-order filters), we need to keep the data at a higher resolution. As argued above, our goal is to preserve the local maxima/minima of the reconstruction, because the trilinear filter cannot interpolate values beyond these limits. In our mixed-resolution building stage, given a set tolerance (the verification/certification stamp) some cells may be classified as subdivision cells. As mentioned above, these cells contain fine details and must be represented by additional data

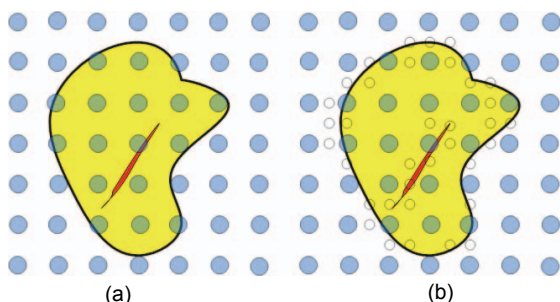


Fig. 7. (a) shows the volume samples at traditional resolution. (b) shows the volume samples with adaptive refinement.

sample points, generated within a progressive refinement process. We store these cells separately as a progressive refinement structure.

Compared to the gold-standard the coarse volume can usually represent the data fairly well and only needs a few locations to refine. The refinement regions contain high frequencies in forms of sharp edges, as pictured in Fig. 7. The high frequencies give rise to a line-shape which is typically sparse in nature. It often occurs across the whole volume which makes brick-boundaries inefficient.

Our mixed-resolution based adaptive refinement has the storage cost of a 3D index volume. The index granularity can be chosen as a certain level of the octree. Each level has a 1/8 smaller storage (1/2 sampling rate in 1D) than its next-lower level. The error determining the local sampling rate is described as:

$$E(x, y, z) = |f(x, y, z) - \sum_{i,j,k \in N} w_{i,j,k} g(x_i, y_j, z_k)| \quad (10)$$

Here, the oversampled reconstruction data  $f(x, y, z)$  is the gold standard,  $w_{i,j,k}$  represent the filter's weights (we use the trilinear filter) and  $g(x_i, y_j, z_k)$  are the grid samples at the coarser level. If the reconstructed signal error compared to the gold standard is larger than the (verified) fidelity threshold  $\epsilon$ , we subdivide the current cell and increase the sampling rate by two. We illustrate our refinement procedure in Fig. 8.

Cell-based and node-based methods are two applicable schemes to represent octree subdivisions. We chose to implement a cell-based octree because the duplicated value on the boundary can overcome the boundary discontinuity problem well. We need the boundary values to preserve the thin structures that often occur in medical datasets (see also Fig. 7). In this cell-based method, the base cell has  $2 \times 2 \times 2$  elements, while the refinement cell could have  $3 \times 3 \times 3$  or  $5 \times 5 \times 5$  elements. A  $2 \times 2 \times 2$  base cell can be subdivided into eight children (a  $3 \times 3 \times 3$  cell) or sixty-four children (a  $5 \times 5 \times 5$  cell). As shown in Fig. 8, our scheme first estimates the error for a  $2 \times 2 \times 2$  cell, which is the traditionally used resolution in CT. If in this cell all the errors (compared to the  $9 \times 9 \times 9$  gold standard) are below the threshold  $\epsilon$ , then we stop and return the refinement index as 0. If there is any estimator reporting an error larger than  $\epsilon$ , we try a  $3 \times 3 \times 3$  cell. When the  $3 \times 3 \times 3$  cell is good then we return a positive index. Otherwise, when the  $3 \times 3 \times 3$  cell fails then we try a  $5 \times 5 \times 5$  cell whose refinement index is always returned as a negative number. We end up with a coarse volume, an index volume and two volume stacks.

The error checking process can be straightforwardly computed on the GPU using 3D textures mapping. The equation to guide the texture coordinate transform is:

$$v_2 \cdot xyz = (v_1 \cdot xyz - \frac{(0.5, 0.5, 0.5)}{l_1} l_1(l_2 - 1)) \frac{l_1(l_2 - 1)}{(l_1 - 1)l_2} + \frac{(0.5, 0.5, 0.5)}{l_2} \quad (11)$$

where  $v_1 \cdot xyz$  is the texture coordinate in a  $l_1^3$  cube and  $v_2 \cdot xyz$  is the texture coordinate in a  $l_2^3$  cube. In our implementation, to better utilize the GPU bandwidth, we process multiple cells simultaneously

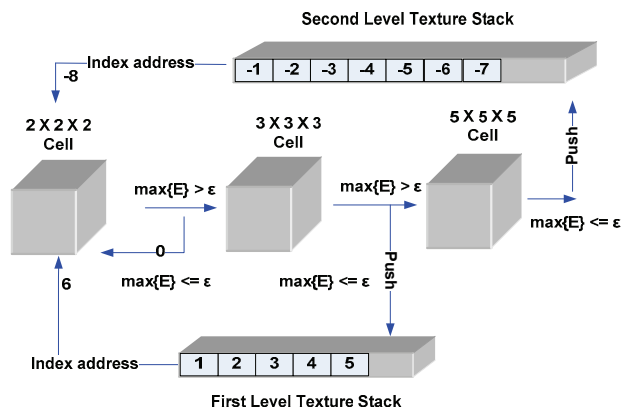


Fig. 8. Adaptive refinement (totally 3 levels) validated by the interpolation filter (trilinear in our case) and the error threshold  $\epsilon$ .

in multiple threads, which can greatly accelerate this pre-processing procedure.

### 4.3 Adding a Continuous Boundary

In the end, there are two texture stacks storing all refinement cells, first level and second level. The element size of these two stacks is  $3 \times 3 \times 3$  and  $5 \times 5 \times 5$ . Along with the  $2 \times 2 \times 2$  cell in the coarse resolution, we have a 3-level mixed-resolution data representation.

Problems in this mixed-resolution representation arise when we are trying to reconstruct values on a mixed-level cell boundary. The trilinear filter itself can preserve C0 continuity across a uniform grid. However, if we have a high resolution cell on one side and low resolution cell on the other, the interpolation scheme would result in a C0 discontinuity. To keep with our verified rendering approach, we need to avoid schemes that blur this T-junction. Instead, we augment the T-junction with the underlying continuous data and upgrade the low-resolution side into a finer resolution. Any cell with high resolution neighboring this cell is affected by it and changes its interface accordingly. Filling the low resolution cell with original data selectively will not dramatically increase the data storage. Most importantly, feeding the gold-standard data will preserve the verifiable threshold in Eq. (10).

The key idea is to only upgrade the high-low resolution boundaries, while the rest of the points are generated by trilinear interpolation (to comply with the low-low resolution boundary). Otherwise the high-low resolution boundary will propagate and we will end up with high resolution everywhere. We illustrate our data feeding process in 2D in Fig. 9. In panel (a), there are eight base  $2 \times 2 \times 2$  cells and one  $3 \times 3 \times 3$  cell with refinement. Each of the 4 neighbors of the refinement cell contains a high-low resolution boundary. In panel (b), we feed the center elements from the gold-standard into the affected cells (with the same blue color). Finally in panel (c), the yellow points are further added into the 4 affected cells. These yellow points are obtained from interpolations and all the T-junctions can be removed. By adding the raw-data rather than blending, our method can adhere to previous error thresholds, and even further increase the authenticity.

The algorithm ensuring a C0 continuous boundary in 3-level overlapping regions is shown in Fig. 9d-g. If the cell is only affected by a  $3 \times 3 \times 3$  region, then data filling occurs similarly as before. If the cell is also affected by a  $5 \times 5 \times 5$  region, we first fill  $3 \times 3 \times 3$  data accordingly (panel (e)), perform another linear interpolation (green samples in panel (f)) and then fill  $5 \times 5 \times 5$  data accordingly (panel (g)).

Our framework performs two processes after building the initial index of data refinement. They are the affected neighbor checking and the data feeding as shown in Fig. 9. The first process scans the initial index and generates maps recording all affected neighbors and the affected boundaries. We use 18 bits to encode different cases because between two neighboring cells, the possible sharing

boundaries can be 6 faces and 12 edges resulting in a total of 18 boundaries. Thus we store 18 bits information for each  $2 \times 2 \times 2$  base cell. Totally there are totally  $18 \times 2 = 36$  bits recording the affected cells by  $3 \times 3 \times 3$  and  $5 \times 5 \times 5$  cells.

The next procedure is filling the low-resolution cell accordingly, as shown in Fig. 9. The previously generated high-resolution data are stored on disk so we do not need to re-generate the original data again. Firstly we look at the cells affected by  $3 \times 3 \times 3$  cell, fill the low-resolution cell accordingly and add those newly upgraded cells into the texture stacks. Then we process the cells affected by  $5 \times 5 \times 5$  cell similar. The added interpolated data of a new  $5 \times 5 \times 5$  cell can be based on interpolating a  $2 \times 2 \times 2$  cell or a  $3 \times 3 \times 3$  cell.

### 4.4 Rendering

While our framework can be generally applied to high order filters, we choose the trilinear filter in this paper for its efficiency, as discussed above. Our fine granular octree framework lends itself quite well to GPU acceleration. We store the base volume and index volume into 3D textures. During ray tracing, each sample position is interpolated and if the current region indicates a finer subdivision then the index points to corresponding children in the octree.

To correctly fetch the index, the index volume should be shifted by  $(0.5, 0.5, 0.5) / volume\_size$  before applying the nearest-neighbor filter. When the ray enters into a refinement cell, the local coordinates will be changed. The texture coordinate mapping between two levels follows the same procedure as the data refinement. So, if one cell has more detail and needs finer resolution, then via the index pointer, the fragment program goes to that position in the corresponding level refinement texture stack to fetch the data. The positive index will be directed to the first level refinement and the negative index will be directed to the second level refinement.

Finally, since our framework serves as an extension of the DVR pipeline, many existing acceleration techniques available to DVR can be incorporated into our pipeline without much modification. We currently use block-based empty space skipping and early fragment kill [10] in our visualization pipeline. But facilitated by our mixed-resolution data representation, we can also readily perform ray-tracing with adaptive step sizes controlled by the local resolution.

## 5 IMPLEMENTATION

We use the FFTW library [15] to perform the Fourier and inverse Fourier transform. The GPU components are based on NVIDIA CG.

There are some implementation issues related to storage and speed. To avoid having to store all high-resolution data in memory at once, we use a block-marching approach which keeps the size of active memory reasonable. The block size should not be too small because the CT reconstruction is done on the GPU [35] (using projective textures) and the bandwidth between GPU memory and CPU memory is relatively low. Therefore we take  $32 \times 32 \times 32$  of  $9 \times 9 \times 9$  cells as a block and reconstruct all of them together. Each block is a  $257 \times 257 \times 257$  floating point data array which takes 64.75 MB of storage. This structure is scalable to large datasets with high resolution. Afterwards, some parameters in our framework can also be fine tuned for better performance. Note that the finer granularities of refinement cells will have more duplicated boundaries. In order to have better storage, we chose the granularity of the refinement to be a factor of 4. Therefore, 64 neighboring  $2 \times 2 \times 2$  cells correspond to one element in the index volume instead of 64 elements. The merged index volume then consumes about 1/64 of the storage of a traditional scalar volume.

In practice the height of the refinement stack could be larger than the current dimension limit of 3D textures. It is often necessary to regroup the two refinement stacks. Under the 4-granular index, the refinement cell would be  $9 \times 9 \times 9$  and  $17 \times 17 \times 17$ . Instead of a  $9 \times 9 \times 9 \times h$  3D volume texture stack, we can regroup the data into  $90 \times 90 \times \lceil 9h/100 \rceil$ . For a  $17 \times 17 \times 17 \times h$  volume stack, we can regroup the data into  $170 \times 170 \times \lceil 17h/100 \rceil$ . The rearrangement of the stack will affect the transform parameters according to Eq. (11).

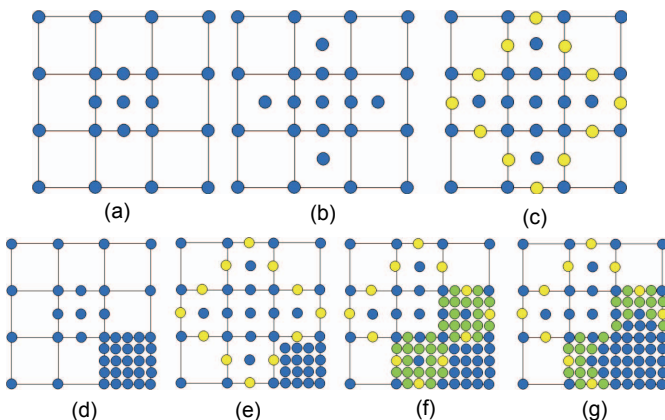


Fig. 9. The data feeding algorithm ensuring a trilinearly interpolated C0 continuous boundary. A 2 level boundary region is shown in (a-c) and a 3 level boundary region is shown in (d-g).

## 6 RESULTS

In this section, we compare results obtained with traditional resolution,  $D^2VR$ , and using our verifiable pipeline. First, we compare the results obtained using the traditional resolution with those from the verified mixed-resolution representation, both based on the verified (frequency-space upsampled) set of projections. Fig. 10 shows a modified Shepp-Logan phantom, with intensity shown as grey values. The image on the left has been rendered at traditional resolution and the one on the right is our verifiable result. In the phantom's definition, the three upper thin tumors have the same intensity as the adjacent smaller but thicker tumor just below (but higher than the three tumors further below). The lower resolution of the traditional method misses the high-intensity peak and thus renders the three thin tumors at a lower intensity than the thicker tumor just below. Our verifiable method, on the other hand, preserves the intensity peaks and visualizes all tumor intensities correctly. Note also the sharper details on strong edges.

The next experiment compares traditional volume upsampling (increasing a volume's resolution via frequency space upsampling) and our verifiable volume oversampling (reconstructing the volume at a higher resolution from upsampled projection data). For this, we simulated a set of 74 X-ray projections (size  $65^2$ ) of the analytical Marschner-Lobb (ML) function [27]. Fig. 11a shows the results of upsampling pipeline: from the projection data, reconstruct a volume at the base resolution of  $64^3$  and then double the resolution to  $128^3$  using frequency-space upsampling (note that this is different from sampling the original 3D ML function and upsampling it, our experiment more closely simulates a practical case – one in which an object is acquired via CT scanning). Conversely, Fig. 11b shows the result of our verifiable pipeline: upsample the projection data by  $8\times$  using the frequency space method and then reconstruct a mixed-resolution volume (2% error threshold) that also has about  $128^3$  data points. We can clearly observe that both renderings are of fairly high quality, but only the verifiable method can represent all function detail (the deep grooves between the rings).

Fig. 12 plots an error map of the results obtained with (a)  $D^2VR$ , (b) DVR (uniform resolution), and (c) VDVR (mixed resolution). For the projection data, both (b) and (c) use frequency space projection upsampling while (a) uses bilinear interpolation as described in [29]. We observe that all pixels in the VDVR panel (c) fall below the set error level of 3%, while without the verifiable mixed-resolution the error increases to 4%.  $D^2VR$  has strong ringing errors in the high frequencies (up to 6%). These error rings cause the 6% RMSE in Fig. 3, and they also result in some missing or reduced high frequency rings in the function (rings 4 and 6 in Fig 13).

We also performed these comparisons using a practical dataset. In clinical or industrial settings the CT scanner X-ray detectors often have higher resolution, and typically a bin decimation procedure is used to remove noise and aliasing. In order to mimic a real-life CT scanning scenario, we performed an  $8\times 8$  down-sampling for decimation. Fig. 14 shows volume renderings of the carp dataset, where 142 projections of resolution  $256\times 129$  each were used for reconstruction to obtain both the traditional volume dataset pictured on the top and the verifiable representation pictured at the bottom. However, the trilinear filter in the X-ray projection simulation would result in strong aliasing, especially along the z-axis. The  $D^2VR$  paper [29] suggests a solution employing a spiral CT simulator. We chose a different route. We first generated an  $8\times$  high-resolution sinogram, added Gaussian noise with density 0.05, and then convolved the projections with a  $5\times 5$  Median kernel and a Gaussian kernel with  $\sigma = (0.1, 4.0)$  before performing an  $8\times$  downsampling. This procedure effectively eliminated the aliasing along the z-axis and also reduced the noise.

Using these simulated projections the dataset was reconstructed and then rendered using a standard trilinear interpolation filter. Fig. 14 panels (a), (d) and (f), were rendered from a uniform  $128^2\times 256$  volume (base) resolution, which is the resolution one would typically pick given the  $256\times 129$  projection data. We observe strong aliasing

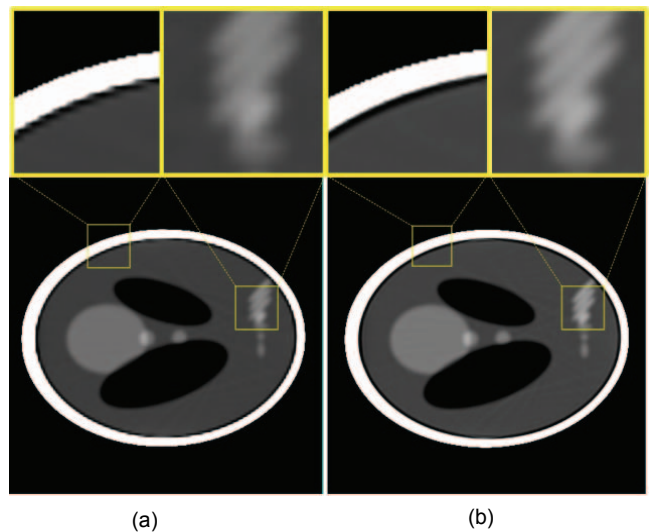


Fig. 10. Shepp-Logan phantom in 2D slice view. The projections were upsampled using the frequency domain method. (a) is based on a uniformly coarse resolution (traditional resolution) and (b) is based on a mixed resolution. Both (a) and (b) use a trilinear filter.

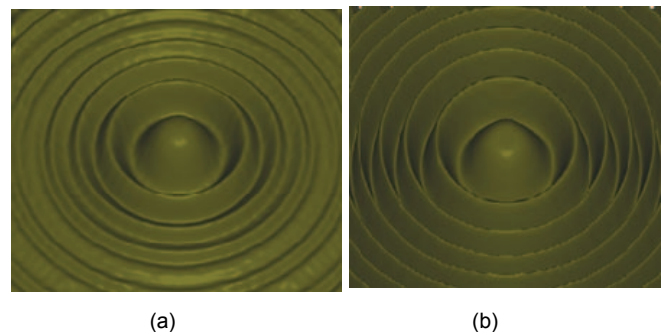


Fig. 11. Iso-surface rendering (using trilinear interpolation) of the ML dataset represented by the same number of volume samples: (a) volume resolution doubled via frequency space upsampling; (b) mixed-resolution reconstructed from upsampled projections and a 2% error threshold.

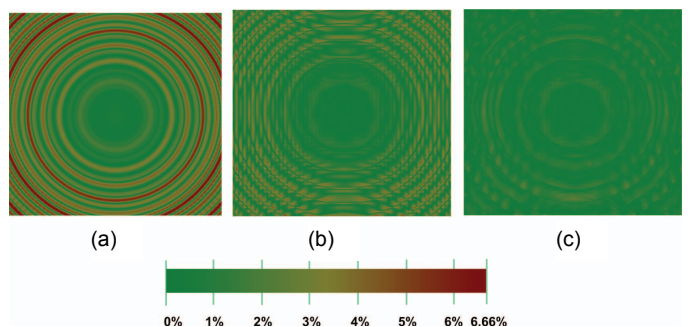


Fig. 12. Error in (a)  $D^2VR$ , (b) DVR at uniform coarse resolution and (c) VDVR using the ML dataset at a 3% error threshold. Projections of (b)(c) are interpolated via the frequency domain interpolation method. (b)(c) use the trilinear filter to interpolate the volume samples.

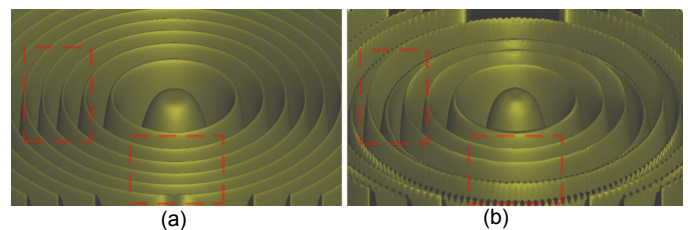


Fig. 13. ML comparison between analytical function (a) and  $D^2VR$  (b).

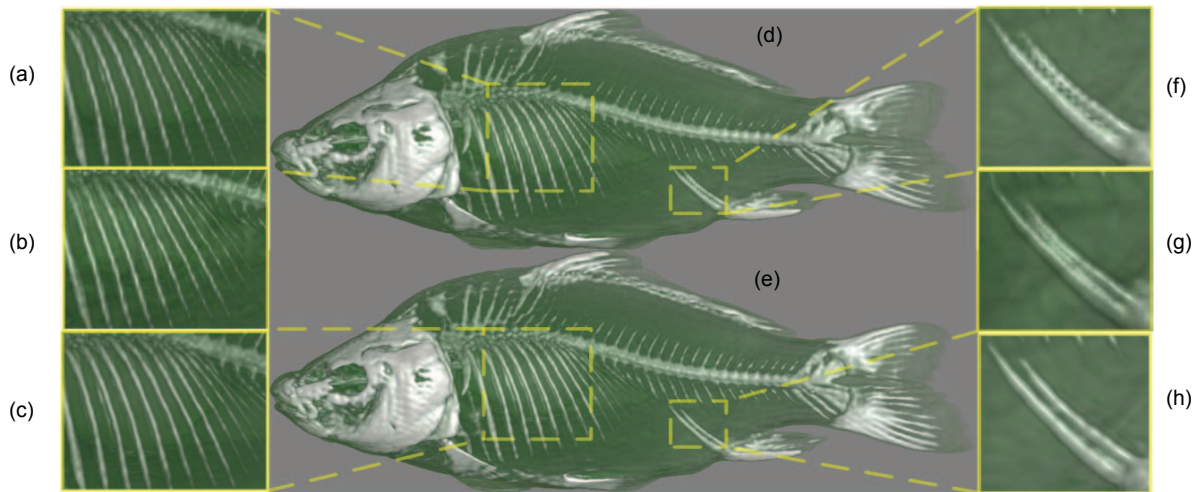


Fig. 14. Renderings of the carp dataset represented in the various grid resolution types. (a)(d)(f) uniform (coarse) resolution, (c)(e)(h) mixed-resolution, (b)(d) frequency-domain upsampled resolution using the same storage than the mixed resolution (magnified cuts only).

artifacts in these renderings. On the other hand, the renderings obtained from the mixed-resolution volume (3% error threshold, 2.4× more storage) and shown in panels (c), (e) and (h) can resolve small detail, such as the thin bones, rather well. We also took the base resolution volume of (d) and used frequency domain upsampling to generate a volume of the same storage than the mixed resolution of (e). Panels (b) and (g) show magnified cuts of a rendering of this volume. These images exhibit similar effects than Fig. 11 – while aliasing is suppressed, small detail cannot be solved. The highlighted fish tail is shown in the beginning of the paper (Fig. 1) which compares our VDVR with D<sup>2</sup>VR. Panel (a) shows that D<sup>2</sup>VR still smoothes out some fine details on the tail, while VDVR in panel (c) provides sharper images, showing the thin bones clearly.

All experiments were performed on a PC with Intel Core 2 Duo 3.00GHz CPU, 1.75GB RAM, and a NVIDIA GeForce 9800 GX2 GPU. Table 2 gives insight into the rendering speed and compares it with D<sup>2</sup>VR. For these results, we rendered into a 512×512 window and used a ray step size of 1. Volume textures were RGBA 32 bit floating point. Central-difference-based gradient estimation was used in the volume rendering. Since the Shepp-Logan dataset has very low contrast tumors and high intensity bones, verification was only performed on intensities around this tissue intensity range.

In Table 1, the error threshold was set to 0.03 for all datasets. We observe that a rendering speed of about 10 frames/s is possible for practical datasets. VDVR requires about 4 times more storage than DVR and likewise D<sup>2</sup>VR. However, we also note that if D<sup>2</sup>VR were to store high-resolution projection data to overcome the aliasing effects incurred from sampling ramp-filtered projection data with an inferior filter, its storage requirements would be significantly higher.

Fig. 15 shows the qualitative effects of the error threshold, using the ML projection data. It appears that refinements will first pick up high-frequency details (the outer rings), and then expand to lower-frequency details (the ring close to the center). Eventually, fine details no longer improve due to the implicit band limit of the ML.

Finally, Table 2 examines the effect of different error thresholds on rendering performance. While the ML function is small, it is rich in fine details. Therefore its storage increases dramatically when the error threshold is lowered. The rendering speed, however, is stable

since the required rendering effort is still small for the GPU. Conversely, the carp dataset is larger in size overall and so requires a larger rendering effort, but it contains only sparse details. Therefore the storage is less sensitive to the error threshold.

### 7 DISCUSSION

The question arises who might benefit from VDVR. Given the traditionally closed architecture of commercial CT scanners, researchers in visualization have no access to raw projection data and so would not be able to generate verifiable volume data themselves. However, an encouraging development in this regard is the currently emerging paradigm shift to flat-panel detector based CT. These panels are likely to produce CT systems that are more open, allowing access to the acquired projection data as well, in addition to the reconstructed volume data. These systems may follow the standard practice of professional SLR cameras that give also access to the raw sensor data, in addition to the processed data.

Further, for simplicity we have only discussed the parallel beam case. Flat-panel detectors, however, produce cone-beam data. Our framework extends quite naturally to this case. First, the corresponding reconstruction algorithm would be the FDK algorithm [14][35]. While the max errors for each 2D projection can be analyzed similarly, depth-weighting factors are now involved in the backprojection. These depth-weighting factors will multiply the projected values and so amplify the errors. Therefore the resulting error bound will be the sum of the max errors multiplied by the max depth-weight factor. The rest of the error analysis factors in the beam geometry but is principally the same than for the parallel beam case.

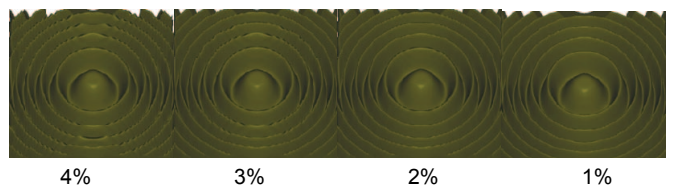


Fig. 15. ML rendered with VDVR at different error thresholds.

Table 1. The Performance of VDVR with 3% error tolerance

Dataset	Base Resolution (# of 2×2×2 cells)	Pre-processing	Rendering	Storage
Shepp-Logan	128×128×128	12 min	10 fps	4×
Carp	128×128×256	35 min	11 fps	2×
Beetle	102×62×128	6 min	13 fps	5×
ML	64 <sup>3</sup>	4 min	20 fps	6×

Table 2. The effects of different error thresholds

Dataset	Threshold	RMS	Rendering	Storage
ML	4%	0.0156	20 fps	2.34×
	3%	0.0123	20 fps	6.04×
	2%	0.00917	20 fps	8.35×
	1%	0.00747	20 fps	32.0×
Carp	3.0%		12 fps	2.41×
	2.5%	NA	11 fps	3.03×
	2.0%		8.6 fps	4.23×

## 8 CONCLUSIONS

We have described an error-aware volume visualization method for data derived from projection-based raw data as occurring in medical, industrial, and security scanning applications. Our method is tightly coupled with the data generation process itself, and it prevents, or at least bounds, the loss of accuracy associated with off-grid sample interpolation during rendering. Our method is able to certify a CT reconstructed volume for use with a given interpolation filter, explicitly specifying the maximum error that might occur in the rendering. It uses a mixed-resolution volume encoding computed in a pre-process. This representation can maintain the advantages of real-time rendering, and at the same time gives the verifiable results.

In our experiments the reconstruction-based error (at  $8\times$  maximum oversampling in our verifiable mixed-resolution volume) was typically less than 1% for real-life dataset while the error threshold in the mixed-resolution volume construction was set to 3%. In contrast, traditional DVR that uses volumes reconstructed at no oversampling (at the projections' resolution) would have maximum errors as high as 20% and more. While one could also oversample the volumes used with DVR, our VDVR will always return a more efficiently sampled volume since it is informed by the theoretical error propagation and the data generation process itself.

On the other hand, the spatial coherence that VDVR maintains allows it to render volumes much more efficiently than  $D^2$ VR which has a (raw projection-based) error bound twice as high (6%) than VDVR (3%). Lowering this error would require an upsampling of the projections, further impeding speed and storage.

Future work will adapt these concepts for more efficient grids, such as BC, and we also plan to evaluate and incorporate other errors occurring in the rendering such as shading, gradient estimation, transfer functions, perception, and the like. Finally, to boost rendering performance, we would like to devise a coarser mixed-resolution scheme — one which would have large sub-volumes of equal resolution that would generate more coherent ray traversals.

## ACKNOWLEDGEMENT

The work was partially supported by NIH grant 5R21EB004099-01 and NSF grant CCF-0702699.

## REFERENCES

- [1] M. Artner, T. Möller, I. Viola, M.E. Gröller, "High-Quality Volume Rendering with Resampling in the Frequency Domain," *Proc. Eurographics/IEEE VGTC Symp. on Visualization*, pp. 85-92, 2005.
- [2] J. Beyer, M. Hadwiger, T. Möller, L. Fritz, "Smooth Mixed-Resolution GPU Volume Rendering," *Proc. IEEE International Symposium on Volume and Point-Based Graphics*, pp. 163-170, 2008.
- [3] R. Bracewell, *The Fourier Transform and its Applications*, 3<sup>rd</sup> edition, McGraw-Hill, 1999.
- [4] L. Condat, T. Blu, M. Unser, "Beyond Interpolation: Optimal Reconstruction by Quasi-Interpolation," *Proc. IEEE. International Conference on Image Processing*, pp. 33-36, 2005.
- [5] C. Crassin, F. Neyret, S. Lefebvre, E. Eisemann, "GigaVoxels: Ray-Guided Streaming for Efficient and Detailed Voxel Rendering," *ACM Symposium on Interactive 3D Graphics and Games*, pp. 15-22, 2009.
- [6] B. Csébfalvi, "An Evaluation of Prefiltered Reconstruction Schemes for Volume Rendering," *IEEE Trans. on Visualization and Computer Graphics*, 14(2):289-301, 2008.
- [7] B. Csébfalvi, B. Domonkos, "Frequency-Domain Upsampling on a Body-Centered Cubic Lattice for Efficient and High-Quality Volume Rendering," *Vision, Modeling, & Visualization Workshop (VMV)*: pp. 225-232, 2009.
- [8] W. Degen, "Sharp Error Bounds for Piecewise Linear Interpolation of Planar Curves," *Computing*, 79(2):143-151, 2007.
- [9] M. do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice Hall, 1976.
- [10] K. Engel, M. Hadwiger, C. Rezk-Salama, J. Kniss, *Real-Time Volume Graphics*. AK Peters Ltd, 2006.
- [11] A. Entezari, T. Möller, "Extensions of the Zwart-Powell Box Spline for Volumetric Data Reconstruction on the Cartesian Lattice," *IEEE Trans. on Visualization and Computer Graphics*, 12(5):1337-1344, 2006.
- [12] A. Entezari, T. Meng, S. Bergner, T. Möller, "A Granular Three Dimensional Multiresolution Transform," *Proc. Eurographics/IEEE-VGTC Symposium on Visualization*, pp. 267-274, 2006.
- [13] T. Etienne, C. Scheidegger, L. Nonato, R. Kirby, C. Silva, "Verifiable Visualization for Isosurface Extraction," *IEEE Trans. on Visualization and Computer Graphics*, 15(6): 1227-1234, 2009.
- [14] L. Feldkamp, L. Davis, J. Kress, "Practical Cone Beam Algorithm," *J. Optical. Society. America A*, 1:612-619, 1984.
- [15] M. Frigo, S. Johnson, "The Design and Implementation of FFTW3," *Proc. of the IEEE*, 93(2): 216-231, 2005.
- [16] L. Hillebrand, R. Lapp, Y. Kyriakou, W. Kalender, "Interactive GPU-Accelerated Image Reconstruction in Cone-Beam CT," *Proc. of SPIE*, vol. 7258, pp. 72582A-72582A-8, 2009.
- [17] R. Kähler, M. Simon, H. Hege, "Interactive Volume Rendering of Large Sparse Data Sets using Adaptive Mesh Refinement Hierarchies," *IEEE Trans. on Visualization and Computer Graphics*, 9(3):341-351, 2003.
- [18] R. Kähler, J. Wise, T. Abel, H. Hege, "GPU-Assisted Raycasting for Cosmological Adaptive Mesh Refinement Simulations," *Proc. Eurographics/IEEE Workshop on Volume Graphics*, pp. 103-110, 2006.
- [19] R. Kirby, C. Silva, "The Need for Verifiable Visualization," *IEEE Computer Graphics and Applications*, 28(5):78-83, 2008.
- [20] E. LaMar, B. Hamann, K. Joy, "Multiresolution Techniques for Interactive Texture-Based Volume Visualization," *Proc. IEEE Visualization*, pp. 355-361, 1999.
- [21] P. La Riviere, J. Bian, P. Vargas, "Penalized-Likelihood Sinogram Restoration for Computed Tomography," *IEEE Trans. on Medical Imaging*, 25(8):1022-36, 2006.
- [22] A. Li, K. Mueller, T. Ernst, "Methods for Efficient, High Quality Volume Resampling in the Frequency Domain," *Proc. IEEE Visualization*, pp. 3-10, 2004.
- [23] P. Ljung, C. Lundström, A. Ynnerman, K. Museth, "Transfer Function Based Adaptive Decompression for Volume Rendering of Large Medical Data Sets," *Proc. IEEE Volume Visualization and Graphics Symposium*, pp. 25-32, 2004.
- [24] P. Ljung, C. Lundström, A. Ynnerman, "Multiresolution Interblock Interpolation in Direct Volume Rendering," *EuroVis*, pp. 259-266, 2006.
- [25] T. Möller, R. Machiraju, K. Mueller, R. Yagel, "Evaluation and Design of Filters using a Taylor Series Expansion," *IEEE Trans. on Visualization and Computer Graphics*, 3(2):184-199, 1997.
- [26] T. Malzbender, "Fourier Volume Rendering," *ACM Trans. on Graphics*, 12(3):233-250, 1993.
- [27] S. Marchesin, G.C. de Verdiere, "High-Quality, Semi-Analytical Volume Rendering for AMR Data," *IEEE Trans. on Visualization and Computer Graphics*, 15(6):1611-1618, 2009.
- [28] S. Marschner, R. Lobb, "An Evaluation of Reconstruction Filters for Volume Rendering," *Proc. IEEE Visualization*, pp. 100-107, 1994.
- [29] P. Rautek, B. Csébfalvi, S. Grimm, S. Bruckner, M.E. Gröller, " $D^2$ VR: High-Quality Volume Rendering of Projection-Based Volumetric Data," *Eurographics/IEEE VGTC Symp on Visualization*, pp. 211-218, 2006.
- [30] J. Smith, P. Gossett, "A Flexible Sampling-Rate Conversion Method," *Proc. IEEE Int'l Conf. Acoustics, Speech, & Signal Proc.*, pp. 112-115, 1984. (Tutorial at <http://ccrma.stanford.edu/~jos/resample>).
- [31] P. Suetens, *Fundamentals of Medical Imaging*, Cambridge University Press, 2002.
- [32] P. Thevenaz, T. Blu, M. Unser, "Interpolation Revisited," *IEEE Trans. on Medical Imaging*, 19(7): 739-758, 2000.
- [33] M. Unser, "Sampling - 50 Years after Shannon," *Proceedings of the IEEE*, 88(4):569-587, 2000.
- [34] F. Xu, K. Mueller, "GPU-Accelerated  $D^2$ VR," *Proc. Eurographics/IEEE VGTC Workshop on Volume Graphics*, pp. 23-30, 2006.
- [35] F. Xu, K. Mueller, "Real-Time 3D Computed Tomographic Reconstruction Using Commodity Graphics Hardware," *Physics in Medicine and Biology*, 52(12): 3405-3419, 2007.
- [36] W. Zbijewski, F. Beekman, "Efficient Monte Carlo Based Scatter Artifact Reduction in Cone-Beam Micro-CT," *IEEE Trans. on Medical Imaging*, 25(7):817-827, 2006.